

UKPEW 2008

24th UK Performance Engineering Workshop

3–4 July 2008
Imperial College London

Ashok Argent-Katwala
Nicholas Dingle
Uli Harder (Eds.)

24th UK Performance Engineering Workshop
3–4 July 2008

Department of Computing
Imperial College London


Edited by
Ashok Argent-Katwala, Nicholas J. Dingle and Uli Harder

Ashok Argent-Katwala, Nicholas J. Dingle and Uli Harder (Eds.)
Department of Computing
Imperial College London
Huxley Building
180 Queen's Gate
London
SW7 2RH

Departmental Technical Reports (Print): ISSN 1469-4166
Departmental Technical Reports (Online): ISSN 1469-4174
ISBN 978-0-9559703-0-6

Published as part of the Technical Reports Series (DTR08-9) of the Department of Computing, Imperial College London, Huxley Building, 180 Queen's Gate, London SW7 2RH.

Copyright of the articles in these proceedings rests with the authors. The proceedings themselves are the copyright of UKPEW (<http://ukpew.org/>).

Picture of Queen's Tower on the front cover by Uli Harder 

Preface

Welcome to UKPEW 2008 at Imperial College London. This is the second time the event has been hosted by Imperial; the last time this happened was 16 years ago in 1992. Other previous locations of UKPEW were:

2007 Edge Hill University	1995 Liverpool John Moores
2006 Poole	1994 Edinburgh
2005 Newcastle	1993 Loughborough
2004 Bradford	1992 Imperial College, London
2003 Warwick	1991 Edinburgh
2002 Glasgow	1990 Bradford
2001 Leeds	1989 Edinburgh
2000 Durham	1988 Edinburgh
1999 Bristol	1987 Edinburgh (Heriot-Watt)
1998 Edinburgh	1986 Edinburgh
1997 Ilkley (Bradford University)	1985 1st UKPEW, Edinburgh
1996 Edinburgh	

This year UKPEW features two keynote speeches: one by Professor Henri Bal from Vrije Universiteit Amsterdam and the other by Adam Grummitt from Metron Technology Ltd. Professor Bal's specialisation is parallel and distributed computing; he has, for example, published work (along with John Romein) on the use of a 144-processor parallel computer to solve the game of Awari, which required the exploration of 889,063,398,406 board positions. Metron Technology Ltd. produces performance management software packages which provide measurement, analysis, planning and reporting capabilities on a wide range of operating systems. Adam Grummitt is currently chair of the UK Computer Measurement Group and is also very active in the IT Infrastructure Library arena.

In total the proceedings include 29 papers from various UK institutions, including Bradford, Edinburgh, Glasgow, Heriot-Watt, Newcastle, Surrey, UCL, Warwick and of course Imperial. In addition to these places there are contributions from Algeria, France, Germany, Holland, Hungary, Iran, Oman, Pakistan and Ukraine. The topics include Grid Computing, Web and E-commerce, Performance Modelling Techniques, Power Management and Wireless Networks.

The social events this year will be a workshop dinner at *Med Kitchen* and before that a trip up the Queen's Tower which will provide some welcome exercise and a fantastic view of the London skyline.

Special thanks go to the "volunteer" referees who very kindly agreed to look through all the original submissions: Soraya Zertal, Felipe Franciosi, Richard Hayden and Fernando Martínez Ortuño. Also, we would like to thank Barbara Claxton, Ann Halford and Teresa Ng who helped with the local organisation.

The programme committee consisting of Ashok Argent-Katwala, Nicholas J. Dingle and Uli Harder doubled up as the local organisers, with Ashok sorting out

the most important items: accommodation and the workshop dinner. They were supported by the conference chairs Jeremy Bradley and William Knottenbelt.

And of course we need to thank the steering committee of UKPEW who gave us the opportunity to hold the event at Imperial College London this year:

Irfan Awan (Bradford)	Stephen Jarvis (Warwick)
Jeremy Bradley (Imperial)	Rob Pooley (Heriot-Watt)
Stephen Gilmore (Edinburgh)	Nigel Thomas (Newcastle)

We also need to thank the EPSRC who gave money to this event through a locally held grant (EP/D061717/1). The Department of Computing at Imperial also supported the event by making the room hire very affordable.

We hope you will all enjoy this year's UKPEW and support the event next year at its 25th anniversary.

London, July 2008
The Workshop Co-chairs & Programme Committee

Contents

1 Performance Prediction and Procurement in Practice: Assessing the Suitability of Commodity Cluster Components for Wavefront Codes <i>by S.D. Hammond, G.R. Mudalige, J.A. Smith and S.A. Jarvis</i>	1
2 Restart in Competitive Environments <i>by K. Wolter and P. Reinecke</i>	18
3 A PEPA Model of a Threshold Policy Sleeping Server <i>by N. Thomas</i>	27
4 A Middleware for Activating the Global Open Grid <i>by J. Cohen, C. Richardson and J. Darlington</i>	38
5 The Performance of Locality-Aware Topologies for Peer-to-Peer Live Streaming <i>by R.G. Clegg, D. Griffin, R. Landa, E. and Mykoniati M. Rio</i>	48
6 Terminating Passage-Time Calculations on Uniformised Markov Chains <i>by A. Clark and S. Gilmore</i>	64
7 Which Battery Model to Use? <i>by M.R. Jongerden and B.R. Haverkort</i>	76
8 Analytical TCP Throughput Model for HSDPA <i>by L. Bodrog, G. Horváth and C. Vulkán</i>	89
9 Rare Event Simulation of Stochastic Activity Networks Using Partition of the Region Technique <i>by A.J. Bidgoly and M.A. Azgomi</i>	107
10 From Architecture to SWN Models for Compositional Performance Analysis of Component Based Systems: Application to CCM based Systems <i>by N. Salmi, P. Moreaux and M. Ioualalen</i>	123
11 Mapping WSLA on Reward Constructs in Möbius <i>by R.Y. Kassab and A. van Moorsel</i>	137
12 Using Representative Intervals for Trace-Based Performance Analysis of Embedded Device Use Cases <i>by L. Pustina, S. Schwarzer and P. Martini</i>	148
13 Fluid Flow Analysis of a Model of a Secure Key Distribution Centre <i>by N. Thomas and Y. Zhao</i>	160

14	Queuing Networks for the Performance Evaluation of Database Designs <i>by R. Osman, I.U. Awan and M.E. Woodward</i>	172
15	Construction of Novel Continuous Time Markovian Queues for Exact Solution <i>by D.J. Thornley</i>	184
16	A System for Dynamic Server Allocation in Application Server Clusters <i>by A.P. Chester, J.W.J. Xue, L. He and S.A. Jarvis</i>	199
17	The Role of Client Behaviour in the Performance Analysis of eCommerce Systems <i>by D.R.W. Holton, I.U. Awan and M. Younas</i>	217
18	Fine Grain Stochastic Modeling and Analysis of Low Power Portable Devices with Dynamic Power Management <i>by Y. Chen, F. Xia, D. Shang and A. Yakovlev</i>	226
19	Distributed Duty-Cycle Management for Dependable Wireless Sensor Networks <i>by J. Wu and Z. Sun</i>	237
20	Validation of Large Zoned RAID Systems <i>by A.S. Lebrecht, N.J. Dingle and W.J. Knottenbelt</i>	246
21	Response Time Distributions via Reversed Processes <i>by P.G. Harrison and M.G. Vigliotti</i>	262
22	Multipath Distance Vector Zone Routing Protocol for Mobile Ad-Hoc Networks MDVZRPA <i>by I.S. Ibrahim, A. Etorban and P.J.B. King</i>	271
23	Guided Subsystem Performance Assessment Architecture for Grid Service <i>by J. Wu, Y. Yang, Y. Zhou, L. Fan and Z. Sun</i>	285
24	RAD Analysis of Adjusted Counter-Based Broadcast in MANETs <i>by S.O. al-Humoud, L.M. Mackenzie, M. Ould-Khaoua and J. Abdulai</i>	300
25	Simulation of a Tandem Router System for Mobile Network Traffic with Different Source Traffic Distributions <i>by A.K. Khan</i>	311
26	On the Componentization of Queue Solution Methods <i>by D.J. Thornley</i>	326
27	Parallelization of a Simulation of a Peer-to-Peer Market for Grid Computing <i>by F. Martínez Ortuño</i>	337
28	State-Space Size Estimation By Least-Squares Fitting <i>by N.J. Dingle and W.J. Knottenbelt</i>	347
29	Hypercube Communication Structures Analysis Via Parametric Petri Nets <i>by D.A. Zaitsev and T.R. Shmeleva</i>	358
30	Capacity Management Views <i>by A. Grummitt</i>	372
31	Large-Scale Parallel Computing on Grids <i>by H. Bal</i>	379

Research Papers

Performance Prediction and Procurement in Practice: Assessing the Suitability of Commodity Cluster Components for Wavefront Codes

S.D. Hammond, G.R. Mudalige, J.A. Smith, S.A. Jarvis,
High Performance Systems Group,
Department of Computer Science,
University of Warwick,
Coventry, CV4 7AL, UK.
{sdh, g.r.mudalige, jas, saj}@dcs.warwick.ac.uk

Abstract

The cost of state-of-the-art supercomputing resources makes each individual purchase an expensive and, in many cases, lengthy process. Often each candidate architecture will need to be benchmarked using a variety of tools to assess potential performance. However, benchmarking alone often provides only limited insight into the potential scalability and suitability of each architecture for procurement.

In this paper we present a case study applying two recently developed performance models to the Chimaera benchmarking code written by the United Kingdom Atomic Weapons Establishment (AWE) with a view to analysing how the code will perform and scale on a medium sized, commodity based InfiniBand cluster. Our models are validated with average accuracies of 90% against an existing InfiniBand machine and then used as the basis for predicting code performance on a variety of hardware configurations including changes in the underlying network, faster processors and high core density per processor.

The results of our experimentation with machine performance parameters demonstrate the compute-bound nature of Chimaera and its sensitivity to network latency at increased processor counts. By using these insights we are able to discuss potential strategies which may be employed during the procurement of future mid-range clusters for a wavefront-code rich workload.

1 Introduction

Modern supercomputing resources are constantly evolving. Where once a ‘super-computer’ may have been a shared memory machine comprising of tens of processors housed in a single structure, today supercomputing resources commonly utilise multiple sub-structures such as cabinets, multiple-processor nodes and more recently multiple-core processors. When combined with the complex network interconnects found in modern systems, identifying and analysing the performance properties of the machine as a whole becomes a significant challenge. With the growing core counts of modern machines and the ever increasing complexity of each system the task of

procuring the ‘right’ computing machinery for purpose is fastly becoming a lengthy and intricate process. Pure benchmarking of applications on candidate architectures serves only limited purpose - the results will only highlight the performance of specific codes and often only for specific inputs. For organisations who want the very best machine performance, a deeper knowledge of code behaviour with respect to each prospective platform is needed.

Performance modelling has been used as a basis for machine comparison [8, 14] and post-installation performance verification [15], and has been shown in a number of examples to address many of the questions which may arise during procurement. Whilst serving as a showcase for many performance modelling techniques, the focus has been on very large emerging architectures and not the small to medium sized commodity or near-commodity clusters used in a number of research organisations. In these procurement activities similar issues must be addressed but with hardware which may have lower specification, be arranged differently or have alternative behaviour to the expensive components that are common place in supercomputing systems.

In this paper we utilise two recently developed performance models to explore the performance of the Chimaera neutron transport benchmark developed and maintained by the United Kingdom Atomic Weapons Establishment (AWE), targetting a processing element count of up to 4096 cores. The direct use and cross-comparison of predictions from two performance modelling techniques aids not only in elucidating specific code and machine behaviour but also in increasing the accuracies of our observations. This work is not intended to comment on the respective costs of each strategy but to provide some degree of quantitative exploration of various hardware and application configurations, which can in turn support the queries that may arise during the early stages of a procurement activity. The specific contributions of this work are:

- The presentation of a performance study for the AWE Chimaera benchmark on commodity or near-commodity hardware. This is the first such study for the Chimaera benchmark and is designed to support future procurement activities for mid-range supercomputing resources at AWE. We use two approaches in verifying our predictions: (1) based on analytic methods utilising the recently developed “plug and play” reusable wavefront model [18] and (2) using a new discrete even simulation toolkit. Both approaches show predictive accuracies of over 90% and provide higher confidence in the conclusions obtained from our performance engineering study.
- A quantitative exploration of the key parameters which affect the performance of wavefront codes on modern commodity HPC systems, supporting the exploration of prospective machine configurations for procurement.
- An exploration of the contention costs arising on a CMP-processor-based cluster when executing Chimaera and the implications for code runtime and machine procurement.

The remainder of this paper is organized as follows, Section 2 provides a brief overview of the two main approaches to application modelling - analytical studies and simulation. We introduce the Chimaera benchmark in Section 3 continuing our discussion in Section 4 by describing the development of two performance models using analytical techniques and a new simulation-based toolkit. Sections 5 and 6 contain our case study in which we benchmark an existing 11.5 TFLOP/s InfiniBand system and project run-times for a variety of alternative application and machine configurations. Our paper

concludes in Section 7 with a summary of the results and a review of the implications for procuring a small to medium size cluster for sustained wavefront-dominated computations.

2 Performance Modelling

Application performance modelling is principally charged with the derivation of models by which code behaviour can be analysed and predicted. In the main, the interest in such models is in analysing how the computational and communication structures in a code will change with respect to an increased processor count or change in application problem size. By developing a deeper insight of the runtime fluctuations resulting from such changes, an understanding of code bottlenecks, software optimisation and in many cases optimal configuration can be developed.

Current techniques for developing application models predominantly fall into two distinct categories - those based on analytical studies and those based on simulation. Although some conceptual work on a binding of the two is discussed in the POEMS framework [1], there has been little practical demonstration reported in academic literature. Analytical studies [11, 13, 21] which seek to represent code behaviour by a series of mathematical formula, are often developed within some modelling framework or methodology (e.g. the LogP[4], LogGP[2] and LoPC[5]). The use of rigid frameworks for modelling helps to alleviate some of the complexity involved in modeling and provide a generic basis upon which code behaviour can be judged. The challenges of using an analytical approach are identifying the key application parameters which affect runtime behaviour and how best to represent each parameter mathematically. The analysis of code for modelling is often based on manual code inspection which, although time consuming, allows the performance modeller to develop a deeper understanding of specific code behaviour from which further behavioural insights may be garnered.

A brief overview of the recently developed “plug and play” reusable wavefront model [18], which serves as the basis for our analytical exploration of Chimaera, is presented in Section 4.1. Note that the development of a reusable model serves to reduce the time required to model future wavefront codes, since a flexible framework can now be applied to any wavefront application; this approach also permits cross-application comparisons to be made within a highly algorithm-specific framework.

Simulation-based performance systems (*e.g.* Wisconsin Wind Tunnel [19], PROTEUS [3] and the PACE toolkit [7, 12]) were originally envisaged as a mechanism to lower the burden of performance modelling by eliminating the need to manually inspect application source code. The automated replay of applications either in source or binary form allowed developers and performance modellers alike to experiment with performance by making direct changes to the application and simulating execution without requiring direct access to the specific machine in question. In practice, the simulation environments developed to date have attempted to directly simulate individual application instructions making the simulation of large industrial codes infeasible in realistic time frames. When the increase of modern application complexity is compounded with increasing core counts of emerging cluster platforms, the use of simulation quickly becomes intractable as a source of fast and efficient performance evaluation. In Section 4.2 we present the development of a prototype simulation toolkit which seeks to overcome some of the problems discussed, in particular the use of coarser grained computational timings (as opposed to individual instructions timings) and a ‘layered’

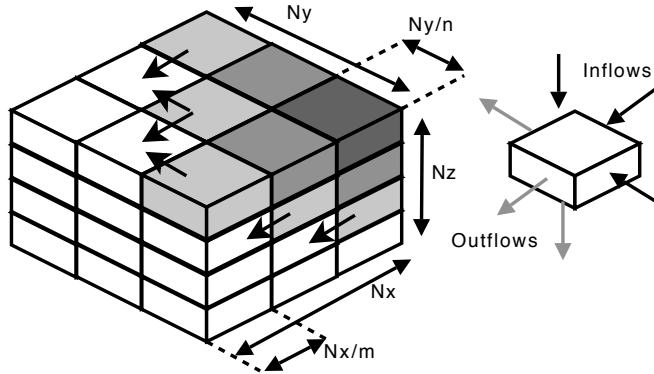


Figure 1: Sweep execution through the data array in Chimaera.

network modelling system, significantly reduce simulation times, whilst providing prediction accuracies commensurate with leading analytical models.

3 The Chimaera Benchmark

The Chimaera benchmark is a three-dimensional neutron transport code developed and maintained by the United Kingdom Atomic Weapons Establishment (AWE). On first inspection the code shares a similar structure with the now ubiquitous Sweep3D application described in numerous analytical performance studies [13, 14, 17]. Unlike Sweep3D, however, the code employs a different internal sweep ordering and utilises a complex convergence criteria to decide when execution is complete. In this section of the paper we present a concise description of the wavefront algorithm employed by both Sweep3D and Chimaera. Our discussion is purposefully brief as a number of existing works describe the behaviour of the wavefront algorithm [16] and a short overview is sufficient to enable an understanding of the key application behaviours.

3.1 The Generic Wavefront Algorithm

The generic three-dimensional wavefront algorithm operates over a data array of size $N_x \times N_y \times N_z$. The data array is decomposed over a two-dimensional processor array sized $m \times n$. Each processor receives a ‘column’ of data sized $N_x/m \times N_y/n \times N_z$. For the purposes of our discussion it helps to consider this column as a stack of N_z tiles each $N_x/m \times N_y/n \times 1$ in size. The algorithm proceeds by executing *sweeps* through the data which pass from one vertex to its opposite. For Chimaera and Sweep3D eight sweeps are used - one for each vertex of the three-dimensional space.

A sweep originates at a vertex of the processor array (the origins of each sweep for Chimaera are shown in Figure 2). The computation required to solve the first tile in the originating processor’s stack is completed and boundary information is exchanged with the two neighbouring processors. Once exchanges are complete the two neighbouring processors solve the first tile in their stack whilst the originating processor solves its second tile. On completion, boundary information is again passed downstream to neighbouring processors. A sweep completes once all tiles in the last processor have been solved. Figure 1 shows a partially complete sweep with dark grey tiles having been solved in previous stages, light grey tiles are currently executing and white tiles

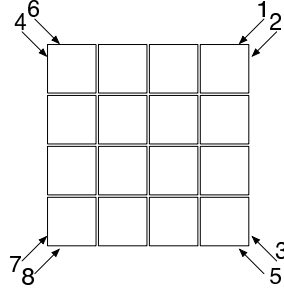


Figure 2: Starting locations for sweep within the two-dimensional processor array employed by Chimaera.

are awaiting boundary information from upstream processors (arrows are used to show visible communications to downstream processors). A full ‘iteration’ of the wavefront algorithm in Chimaera requires all eight sweeps to have completed.

4 Modelling Chimaera

The modelling of Chimaera has been conducted using two approaches - analytical modelling based on the “plug and play” reusable model [18] and using the new WARwick Performance Prediction (WARPP) simulation toolkit developed by the University of Warwick.

4.1 Plug and Play Analytical Model

Model Parameter	Chimaera Value
N_x, N_y, N_z	Input size
W_g	<i>measured</i>
$W_{g,pre}$	0
$H_{tile}(cells)$	1
n_{sweeps}	8
n_{full}	4
n_{diag}	2
Message Size _{EW} (Bytes)	$8H_{tile}$ $\times \#angles$ $\times N_y/m$
Message Size _{NS} (Bytes)	$8H_{tile}$ $\times \#angles$ $\times N_x/n$

Table 1: Reusable Wavefront Model Application Parameters.

The “Plug-and-play” reusable wavefront model developed in [18] represents the culmination of three individual application performance studies for the Sweep3D, Chimaera and NAS-LU benchmarks. By using the insights obtained in modelling these three wavefront codes, Mudalige, Vernon and Jarvis have extracted and abstracted the common parameters (shown in Table 4.1) which affect application runtime into a generic model. The computational time required, W_g , and the computational time per cell prior

$W_{pre} = W_{g,pre} \times H_{tile} \times N_x/n \times N_y/m$	(r1a)
$W = W_g \times H_{tile} \times N_x/n \times N_y/m$	(r1b)
$StartP_{1,1} = W_{pre}$	(r2a)
$StartP_{i,j} = \max(StartP_{i-1,j} + W_{i-1,j} + Total_comm_E + Receive_N,$ $StartP_{i,j-1} + W_{i,j-1} + Send_E + Total_Comm_S)$	(r2b)
$T_{diagfill} = StartP_{1,m}$	(r3a)
$T_{fullfill} = StartP_{n,m}$	(r3b)
$T_{stack} = (Receive_W + Receive_N + W + Send_E + Send_S + W_{pre})N_z/H_{tile} - W_{pre}$	(r4)
$Time\ per\ iteration = n_{diag}T_{diagfill} + n_{full}T_{fullfill} + n_{sweeps}T_{stack} + T_{nonwavefront}$	(r5)

Table 2: Plug-and-play LogGP Model: Single Core Per Node.

to the algorithm kernel, $W_{g,pre}$, are the only machine specific values for which benchmarking of the application is required.. For our study this was obtained by using a manually instrumented version of the benchmark which times the core computational kernel of the wavefront algorithm. $W_{g,pre}$ is unused in Chimaera since there are no computational sections in the sweep algorithm prior to the main kernel.

The sweep ordering parameters, n_{sweeps} , n_{full} and n_{diag} represent the total number of sweeps per iteration, the number of full sweeps and the number of half sweeps respectively. The concept of ‘full’ and ‘half’ sweeps relates to the ability of sweeps within the application to overlap. Recall the sweep ordering presented in Figure 2. Sweep 2 originates on the processor located in the top right corner of the processor array. Once this sweep has successfully passed through the bottom right (the starting location for sweep 3) the next sweep can begin. If this starts prior to sweep 2 finishing on the bottom left processor, overlapping occurs which serves to increase the efficiency of the code. Overlapping can only occur if sweep i finishes at the starting location for sweep $i + 1$ whilst other downstream processors are still processing sweep i . This occurs twice in Chimaera (sweep pairs 2,3 and 6,7) giving an n_{diag} value of 2. The full reusable model is presented in Table 2 with the complete equation for runtime given in (r5). Explanations of each sub-equation are given in [18]. Note that in the original paper describing the reusable wavefront model, the authors develop a complex LogGP communications model for the Cray XT4 architecture. In this work we develop a simpler regionalised least squares regression model to obtain times for MPI send and receive operations (these are presented in Section 5.1.1).

4.2 Simulation using the WARPP Toolkit

The WARwick Performance Prediction (WARPP) toolkit presented in this paper is a prototype performance prediction toolkit and evaluation engine, which has been designed to support performance prediction and code analysis on machines containing thousands of processors. More specifically, we intend for our toolkit to provide accurate simulations for modern Massive Parallel Processor (MPP) machines which might consist of multi-core, multi-processor cabinet structures each having their own complex interconnect or protocol. As the sizes of future machine architectures to continue to grow, we expect that additional sub-structures will be required to support increasing core counts, again each is likely to have its own performance properties adding further complexity to modelling activities. With this in mind the structure of a machine is relayed to the simulator by a series of ‘profiles.’ Each profile has unique performance properties such as network latency, outbound bandwidth etc. When developing a sim-

ulation the user is required to specify the respective values for each performance property and a mapping of MPI processes to profiles for the specific machine configuration being analysed. By providing a generic basis for the description of a machine, arbitrarily complex hardware models can be developed enabling the exploration not only of modern machine structures but also future multi-structured computing resources.

Simulations developed using the WARPP toolkit build upon the observation that parallel codes are ordered executions of basic blocks separated by control flow, calls to network transmissions or I/O operations. Like previous simulators we recreate application behaviour by replaying the code's control flow, pausing during execution to directly simulate computation, communication and I/O events. Communication between processes are simulated fully ensuring that transmissions between nodes block when the transmissive partner is otherwise engaged. Computation is, however, modelled quite differently to existing work in that it does not simulate each application instruction directly. Instead, the toolkit jumps over whole basic blocks within the control-flow recording the time that the block requires for execution on the target platform. The switch to coarser grained computational timings significantly reduces the time required for individual simulations aiding in improving the scalability of the simulator to considerably higher processor counts than previous toolkits. The issue which arises in moving to coarser grained computational timings is precisely how the time for the block is extracted from the application. To alleviate the manual instrumentation of code to obtain such timings the toolkit includes an automated code analyser which injects timing routines into the application source code directly creating an instrumented benchmark version of the code. The analyser also generates a control flow representation of the code detailing where each block can be found and how to identify its associated execution time from the instrumented application output.

4.2.1 Developing a Simulation in WARPP

Developing a WARPP simulation involves three stages. In the first the application source code is analysed using automated code analysis tools - these are responsible for diagnosing the 'basic blocks' of the application and extracting a control flow graph for each process in the parallel application. Basic blocks are considered to be separated by either a change in the address counter (as would be caused by a branching statement or loop) or a communication (such as an `MPI_Send` or `MPI_Recv`). Once the basic blocks have been found, each is instrumented with timing routines to record the wall time that is required for execution. Two outputs are produced at this stage of simulation - an instrumented version of the application's source code and a basic performance model which describes the control flow of the application, the arrangement of basic blocks within this control flow and the points at which communication and I/O occurs.

The second stage of simulation requires the user to benchmark the machine using the instrumented version of the code and some reliable MPI benchmarking utility (such as `MPPTest` [22] or the `Intel MPI Benchmark` [9]). The output of these benchmarks, which takes the form of a 'work time' for each sequential block and a set of network latencies and bandwidths, is then fed into the third stage of simulation where the control flow is replayed using the wall clock times of each block to calculate the compute resources required and the communication points in the application directly simulated to obtain a communication model.

During a simulation, data relating to the application's performance and machine utilisation is recorded enabling performance modellers to replay the simulated execution at a later date and analyse where execution time was spent (for example, time spent

Network Profile	Message Size (Bytes)	n_l (microseconds) Value	B (GBytes/s) Value
on-chip (core to core)	≥ 0	0.655	2.70
off-processor (processor to processor)	< 2048	0.69	2.80
	≥ 2048	0.91	3.83
off-node (node to node)	< 2048	2.64	0.46
	≥ 2048	3.63	0.73

Table 3: Benchmarked Network Performance for the CSC-Francesca Machine (measurements taken using the Intel MPI benchmarking utility version 3.0 [9]. The Intel C compiler version 10 was used with default system MPI libraries.)

Core Count	Problem Size	Actual Runtime (s)	Analytical Pred. (s)	Simulation Pred. (s)	Analytical Error (%)	Simulation Error (%)
32	120 ³	107.18	88.76	89.58	-17.19	-16.42
64	120 ³	56.72	47.59	48.75	-16.09	-14.04
128	120 ³	32.56	28.20	28.98	-13.40	-11.01
81	240 ³	342.33	326.45	330.46	-4.64	-3.47
96	240 ³	297.03	268.71	277.56	-9.54	-6.55
100	240 ³	278.37	243.36	248.32	-12.58	-10.79
128	240 ³	225.65	205.50	207.18	-8.93	-8.18
169	240 ³	174.35	174.35	177.09	-0.88	1.57
256	240 ³	129.65	115.58	117.98	-10.85	-9.01

Table 4: Model Validations on the CSC-IBM Francesca Machine - (Compiler - Intel Fortran 10.0 with -O2 optimisation setting, OpenMPI 1.2.5, All runtimes given are wall time for sweeping components in seconds, Negative values indicate under-predictions)

in communication, computation, idle etc). As our studies into the applications used at AWE deepen we intend to use the simulation data to direct potential improvements in code structure and resource allocation.

5 Modelling Code Performance on a Commodity High Performance Cluster

In this section we present the results of a benchmarking and modelling exercise conducted on the recently installed Centre for Scientific Computing (CSC) *Francesca* machine operated by the University of Warwick. The benchmarked values from this machine serve two purposes - firstly to allow us to verify our performance models against a set of known runtimes ensuring accuracy, and secondly to form the basis of projections for alternative machine configurations that may be considered during a procurement exercise.

5.1 The University of Warwick Centre for Scientific Computing (Francesca) Machine

The recently installed 11.5 TFLOP/s Centre for Scientific Computing (University of Warwick) IBM supercomputer is typical of a large, sub-Million pound commodity cluster available today. The system comprises of 240 dual-Intel Xeon 5160 dual-core nodes each sharing 8GB of memory (giving 1.92TB in total). Nodes are connected via a QLogic InfiniPath 4X, SDR (raw 10Gb/s, 8Gb/s data) QLE7140 host channel adapters (HCAs) connected to a single 288-port Voltaire ISR 9288 switch. Processor core to HCA ratio is 4 : 1. Each compute node runs the SUSE Linux Enterprise Server 10 operating system and has access to the IBM GPFS parallel file system [20]. For our study the Intel C/Fortran 10 compiler suite was used in conjunction with OpenMPI 1.2.5 [6] and the PBS Pro scheduler. By default, jobs launched under PBS are allocated ‘freely’ in the system - *i.e.* to any free core which meets the wall time or memory resources requested by the job. Nodes and processors are shared between jobs unless specifically requested during submission. Runtimes can therefore vary (by as much as 10-15%) between successive runs due to the ‘free’ placement of processes within the machine and the potential sharing of node resources.

5.1.1 Machine Network Benchmarks and Models

The results of machine benchmarking demonstrating raw MPI latency and bandwidths are shown in Table 3. Note that the network benchmarking is partitioned into two regions by message size. The point at which the split in network performance occurs is 2048 bytes, indicating that the InfiniBand management system may be configured for a maximum transmission unit (MTU) size of 2Kbytes (a maximum of 4K is supported by the HCA and switch).

For both performance studies we model the communication time for a message of length x bytes as $t_{send}(x) = (1/B)x + n_l$ with the bandwidth (B) and latency (n_l) associated with the appropriate region for x . The time for a receive is modelled by: $t_{recv}(x) = (1/B)x$ since the receiver does not experience the latency required to establish the connection but must spend at least the actual transmission time in a locked state accepting data from the network interconnect. Using these values we can calculate the point at which bandwidth will dominate network transmissions as: $(2.62 \times 10^{-6})/(1/(0.46 \times 1024^3)) = 1304$ bytes (small messages) and $(3.63 \times 10^{-6})/(1/(0.73 \times 1024^3)) = 2846$ bytes for large messages. In the context of Chimaera these values, where each cell contains 10 angles, each of which is a double floating point value, equate to message sizes of 17 and 36 cells respectively. These values indicate the “see-saw” point at which the network operates, giving some indication of whether bandwidth or latency is dominant for each MPI operation.

5.2 Performance Model Validation

Table 4 presents validations of both performance models for the CSC-Francesca machine. The average prediction error is 10.46% for the analytical model and 9.03% for the simulation demonstrating the high degree of accuracy in the models and the strong correlation between both studies.

Note that the vast majority of the predicted runtimes are below the actual execution time - the principle reason being that both performance models assume as ‘perfect’ allocation of processor cores within the machine, assuming that neighbouring MPI

ranks will be allocated as closely as physically possible. In practice, the free placement of processes causes some degree of increased execution time due to the higher network costs experienced. Similarly, the natural load and noise which occurs from shared resources helps to create variation in execution. Additionally, predictions are taken from averaged estimates of machine parameters for which rounding and measurements may also occur.

6 Procurement: Assessing the suitability of machine components

Following the benchmarking of the CSC-Francesca machine and validation of the performance models, we present several sub-studies exploring alternative machine or application configurations. In the following studies we analyse the effect on code runtime of a change in (1) an increase in problem size, (2) moving to a gigabit ethernet networking solution (3) the installation of InfiniBand resources with identical bandwidth but increased latency (4) a change in the performance of individual processor-cores and (5) a doubling of processor-core density.

6.1 Large Problem Sizes

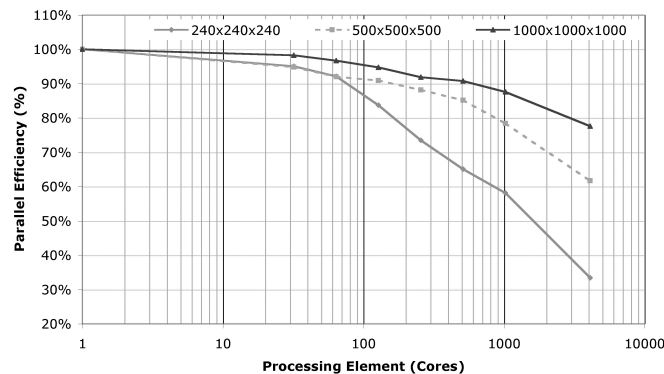


Figure 3: Parallel Efficiency of Large Problem Sizes using the InfiniBand Interconnect.

New computing machinery is often purchased with the intention of not only running current codes but also future higher complexity problems or larger input sizes. The decision of which machine to purchase today may often be governed by expectations of how future users intend on using the system. Figure 3 presents the expected parallel efficiency of an increased input size with increasing processor count. Note that there is a significant decline in efficiency for each input size as the PE count rises. This effect is attributable to the increasing proportion of runtime accounted for by communication resulting from decreasing computation time per processor and an increase number of network transmissions in the system as a whole.

The measure of parallel efficiency is of particular interest to AWE since parallel jobs are mandated to be in higher than 50% configurations wherever possible with a number of users specifically choosing PE counts to target this value. For the 240³

problem this point occurs between 1024 and 2048 cores indicating the approximate core count which may be required per job if targeted specifically for a 50% efficiency. Depending on how many simultaneous jobs the organisation wants to execute at this level of efficiency an approximate core count for procurement can be deduced. For larger problem sizes a similar form of analysis is also applicable, however, significantly more cores will be required before the 50% point is reached.

6.2 Choice of Networking Interconnect

For any machine intended to execute high performance parallel codes the choice of interconnect is particularly accute. The precise mix of latency, bandwidth capacity and cost must be balanced to support the compute resources in delivering smooth, consistent performance. At the time of procurement it is common to want to assess not only which interconnect will provide the best raw performance but also what the effect of changing the interconnect or choosing a slightly lower specification will have on overall runtime. We have modelled two such choices - (1) whether to select a Gigabit network over an InfiniBand interconnect and (2) the effect of purchasing an InfiniBand network with identical 4x, SDR bandwidths but 25%, 50% and 75% higher latencies.

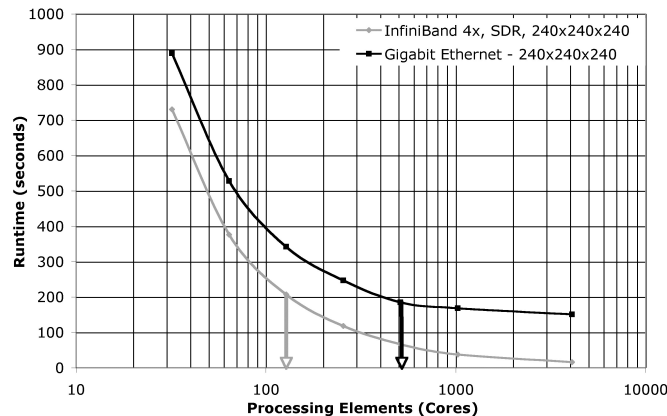


Figure 4: Chimaera Runtime using InfiniBand (4x, SDR) and Gigabit Ethernet Interconnects.

Figure 4 presents the predicted runtimes for a hypothetical machine in which we have replaced the InfiniBand interconnect with a gigabit ethernet network. The gigabit runtime is consistently over 100 seconds slower than the InfiniBand system reflecting the impact of increased latency and a significant decrease in bandwidth. In analysing the results we propose that the reader considers the economics of purchasing either fewer processors and a more expensive InfiniBand network or a greater number of processors and a less expensive gigabit interconnect - a typical decision which may be faced in any procurement activity. For the Chimaera benchmark at least, the results demonstrate that almost twice as many processors will be required to offset the degradation of using a slower interconnect - a significant increase which will in turn make the machine more expensive to run and potentially more difficult to administer.

In Figure 5 we demonstrate predictions for the percentage increase in runtime resulting from the use of an 4x SDR InfiniBand interconnect with 25%, 50% and 75% higher latencies. For small processor counts (less than 1000) the increase in runtime is

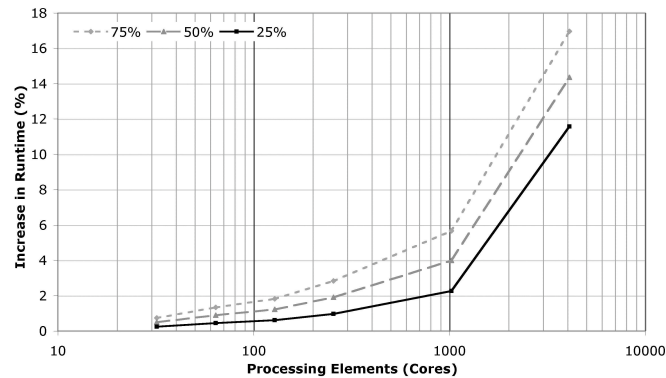


Figure 5: Increase in runtime from a 4x SDR InfiniBand network with varying increases in latency.

less than 6% in all cases. After this point - where communication begins to become a higher proportion of runtime - the runtime begins to increase rapidly with an increase of at least 10%. In this scenario the purchase of a lower specification system may be acceptable if the intention is to limit the maximum processor count of each job to 1024 cores or less.

We also believe that the use of machine configurations for node counts greater than 288 will cause increases in experienced wire latencies as tree based switch topologies will need to be employed in order to cope with the extra port count. These costs are not included in this work as benchmarked values to support a predictive model are not currently available and work completed in [10] provides some suggestion that contention within InfiniBand switches may be reduced in future systems through the use of advanced routing algorithms. Figure 5 does however help to give indication of how sensitive the structured communication pattern used in Chimaera is to even minor increases in network latency.

6.3 Machine Compute Performance

The compute resources of the machine are usually the feature which draws the most attention. Whilst only part of the picture for parallel systems, the computational aspects of a code are often better understood by domain experts and developers. With increasing variation in processors being offered in the form of increasing core counts and arrangements, considerable clock speed differences and in some cases, varying cache implementations, choosing the ‘right’ processor for an application can be difficult. We present several studies in this section of the paper which attempt to quantify the performance benefit of choosing either 10% or 20% faster processors, 10% slower processors or making the move from the existing dual-core Intel Xeon 5160 processors to quad-core chips with the same per-core performance but high core-density per processor.

6.3.1 Increased Individual Core Performance

Figure 6 presents the predicted change in runtime from using dual core processors with individual core performances of +10%, +20% and -10%. The diminishing returns

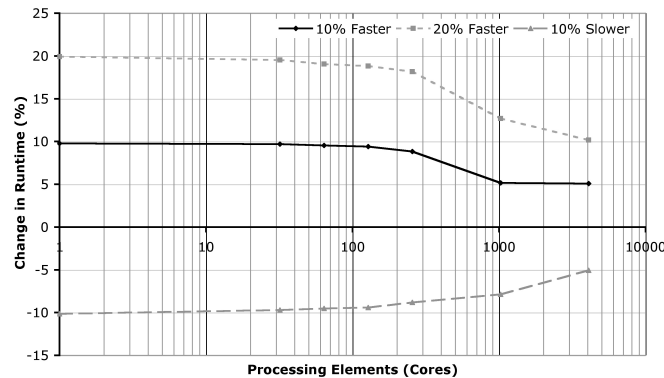


Figure 6: Increase in runtime from a varying changes in individual processor-core performance.

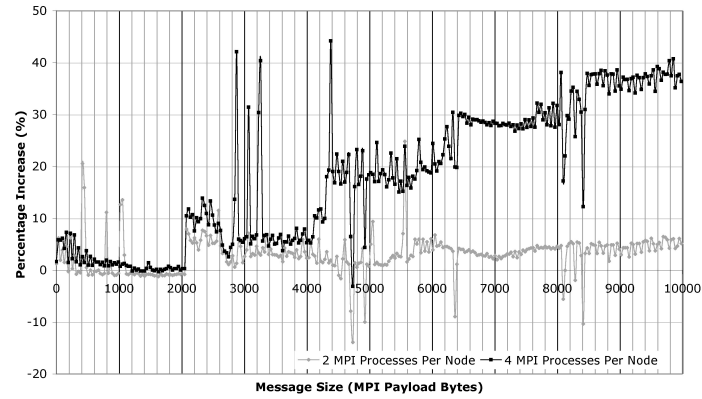
demonstrate the respective points at which communication begins to dominate runtime. In each case the change in runtime performance is approximately equal to the change in per core performance for small processor counts. As the processor count rises the impact on runtime is reduced due to the increase proportion of runtime accounted for by communication, reducing the contribution of faster computational resources to the runtime. Note that at increased processor counts the impact on runtime of using a slower processor is also reduced. The choice of core performance should therefore be considered in the context of job size - at small job sizes the runtime is improved best by using the fastest processors possible, as the core count in use rises there are diminishing returns from employing faster computational resources.

6.3.2 Increased Core Density - Dual versus Quad Core

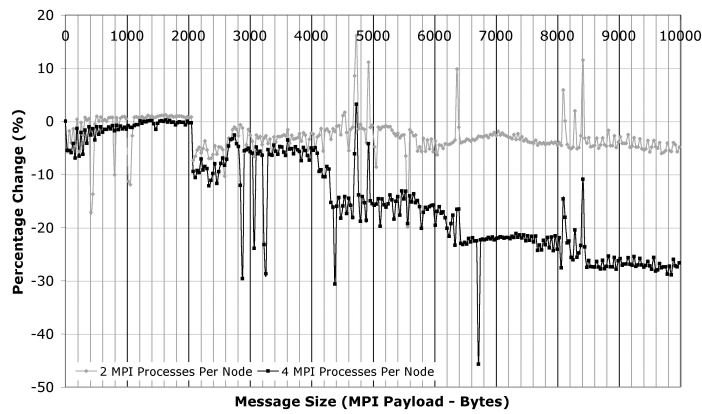
With an increasing variety of multi-core processors becoming available including dual, quad and oct-core configurations, a common issue arising in procurement is which core density to select in designing the machine's compute architecture. On initial consideration the economic advantages of higher core densities are consolidation and reduced power or cooling demands per core, however, the increasing density often impacts on runtime performance.

In Figures 7(a) and 7(b) we show a set of results obtained from running the Intel MPI benchmark in three configurations - one, two and four MPI processes per node respectively. The increasing number of processes per node (which is the effect of higher core densities) reduce the per-core network performance. The increased time to perform an MPI send, and the decreased per core bandwidth, result from high levels of contention for the single InfiniBand HCA per node. Each process must wait longer before having exclusive access to the machine network. If core densities continue to rise then this will continue to impact performance unless the issue of contention is addressed by increasing the number of networking channels per node - the economic effect of this may be a significant addition to procurement cost.

We have modelled the effect on runtimes of replacing each existing dual-core processor with a quad-core equivalent in which the per-core performance of the chip remains identical. The network latency for the InfiniBand network has been left the same for message sizes less than 2048 bytes, increased by 10% for message sizes ranging



(a) Percentage change in time required to complete MPI send operation.



(b) Percentage change in per-core network bandwidth

Figure 7: Percentage change in time to send and per-core bandwidth when increasing the MPI process per node from one to two and four.

Total Core Count	Dual Core Runtime (s)	Quad Core Runtime (s)	Percentage Change (%)
32	729.97	726.19	-0.52
64	376.46	373.62	-0.75
128	207.18	207.92	0.36
256	117.98	118.50	0.44
1024	66.64	66.33	7.88
4096	37.29	40.45	8.46

Table 5: Predicted Quad versus Dual Core Performance (The quad-core configuration is modelled with an increased in time to send and reduced bandwidth to account for contention).

from 2048 to 4096 bytes and increased by 20% for larger messages. Network bandwidth has been changed by the same value but decreased. The changes in latency and bandwidth are drawn from the observed values shown in the figures above. Table 5 presents our predicted runtimes for the quad core machine compared with the existing dual core structure. Initially performance is improved since there are more cores utilising the fast core-to-core transmission speeds. Once core counts reach 1024 processors the increased latency and reduced bandwidth create up to an 8% increase in runtime.

7 Conclusions

In this paper we have presented a case study detailing the application of two performance models - one based on analytical techniques and the other based on simulation - in supporting the procurement of a large, sub-Million pound commodity cluster for a wavefront-code rich workload. The study explores the performance and scalability of the Chimaera benchmark code used by the United Kingdom Atomic Weapons Establishment.

We demonstrate average predictive accuracies of 90% for a variety of processor configurations and input sizes. The cross-correlation of predictions from two contrasting performance models serves to increase the confidence in our predictions and the insights obtained during our subsequent analysis.

More specifically, this paper shows:

- Quantitative estimates for the parallel efficiency of existing and future problem sizes that are of interest to AWE;
- That a system with a low performance network will require a greater processor count to offset the effect of higher latencies and lower bandwidth. We demonstrate this by projecting the performance of a Gigabit ethernet network in comparison to a faster InfiniBand system, showing approximately twice as many processors are required by the ethernet system to achieve comparable levels of performance at core counts less than 1024;
- Improving/reducing the latency performance by a factor of 2, results in up to 10% change in overall runtime;
- For small processor counts the overall runtime varies by the factor of improvement in per-core performance, but as core counts increase, the contribution of faster per core performance provides diminishing returns;
- Increasing the core density per processor reduces the performance due to contention for memory and network resources. We estimate the quantitative degradation of overall runtime when doubling core-density from dual to quad core processors to be approximately 8% up to 4096 cores on the commodity InfiniBand system studied.

Our results demonstrate that the selection of machine configuration and processor count should be directed by the average size of jobs the machine is intended to execute. For multiple small jobs, individually faster processors should be prioritised over a faster interconnect, since the code is predominantly compute bound at these points. For larger

jobs, the interconnect plays a more significantly role in performance indicating that a more expensive, low latency network should be targetted during procurement.

The predictive models used in this study demonstrate efficient, low-cost and rapid methods to gather quantitative and qualitative insights into questions which arise during procurement for both currently available and future systems. In contrast, traditional approaches such as direct benchmarking require significant and expensive machine execution time and more effort to arrive at a subset of conclusions limited solely to currently available machine configurations.

Acknowledgements

Access to the Chimaera benchmark was provided under grants CDK0660 (*The Production of Predictive Models for Future Computing Requirements*) and CDK0724 (*AWE Technical Outreach Programme*) from the United Kingdom Atomic Weapons Establishment. Access to the CSC-Francesca machine was provided by the Centre for Scientific Computing at the University of Warwick with support from the Science Research Investment Fund.

References

- [1] V.S. Adve, R. Bagrodia, J.C. Browne, E. Deelman, A. Dubeb, E.N. Houstis, J.R. Rice, R. Sakellariou, D. Sundaram-Stukel, P.T. Teller, and M.K. Vernon. POEMS: End-to-end performance design of large parallel adaptive computational systems. *Software Engineering*, 26(11):1027–1048, 2000.
- [2] A. Alexandrov, M.F. Ionescu, K.E. Schauser, and C. Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1997.
- [3] Eric A. Brewer, Chrysanthos Dellarocas, Adrian Colbrook, and William E. Weihl. PROTEUS: A High-Performance Parallel-Architecture Simulator. In *Measurement and Modeling of Computer Systems*, pages 247–248, 1992.
- [4] D.E. Culler, R.M. Karp, D.A. Patterson, A.Sahay, K.E. Schauser, E. Santos, R. Subramonian, and Thorsten von Eicken. LogP: Towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.
- [5] Matthew Frank, Anant Agarwal, and Mary K. Vernon. LoPC: Modeling Contention in Parallel Algorithms. In *Principles Practice of Parallel Programming*, pages 276–287, 1997.
- [6] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.
- [7] G.R. Nudd, D.J. Kerbyson, E.Papaefstathiou, J.S. Harper, S.C. Perry and D.V. Wilcox. PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems. *The International Journal of High Performance Computing*, 4:228–251, 2000.
- [8] A. Hoisie, G. Johnson, D.J. Kerbyson, M. Lang, and S. Pakin. A Performance Comparison through Benchmarking and Modeling of Three Leading Supercom-

- puters: Blue Gene/L, Red Storm, and Purple. In *Proc. IEEE/ACM SuperComputing*, Tampa, FL, October 2006.
- [9] Intel Corp. MPI Benchmark Utility. 2008.
- [10] G. Johnson, D.J. Kerbyson, and M. Lang. Optimization of InfiniBand for Scientific Applications. In *International Parallel and Distributed Processing Symposium 2008 (IPDPS'08)*, Miami, Florida, USA, April 2008.
- [11] Richard M. Karp, Abhijit Sahay, Eunice E. Santos, and Klaus E. Schauer. Optimal broadcast and summation in the logp model. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 142–153, 1993.
- [12] D. J. Kerbyson, J. S. Harper, A. Craig, and G. R. Nudd. PACE: A Toolset to Investigate and Predict Performance in Parallel Systems. In *European Parallel Tools Meeting, ONERA, Paris*, October 1996.
- [13] D.J. Kerbyson, A. Hoisie, and S.D. Pautz. Performance Modeling of Deterministic Transport Computations. In *Performance Analysis and Grid Computing*, Kluwer, October 2003.
- [14] D.J. Kerbyson, A. Hoisie, and H.J. Wasserman. A Comparison between the Earth Simulator and AlphaServer Systems using Predictive Application Performance Models. *Computer Architecture News (ACM)*, December 2002.
- [15] D.J. Kerbyson, A. Hoisie, and H.J. Wasserman. Use of Predictive Performance Modeling During Large-Scale System Installation. In *Proceedings of PACT-SPDSEC02*, Charlottesville, VA., August 2002.
- [16] L. Lamport. The Parallel Execution of DO Loops. *Commun. ACM*, 17(2):83–93, 1974.
- [17] G.R. Mudalige, S.A. Jarvis, D.P. Spooner, and G.R. Nudd. Predictive Performance Analysis of a Parallel Pipelined Synchronous Wavefront Application for Commodity Processor Cluster Systems. *Cluster 2006*, 2006.
- [18] G.R. Mudalige, M.K. Vernon, and S.A. Jarvis. A Plug and Play Model for Wavefront Computation. In *International Parallel and Distributed Processing Symposium 2008 (IPDPS'08)*, Miami, Florida, USA, April 2008.
- [19] S.K. Reinhardt, M.D. Hill, J.R. Larus, A.R. Lebeck, J.C. Lewis, and D.A. Wood. The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers. In *Measurement and Modeling of Computer Systems*, pages 48–60, 1993.
- [20] Frank Schmuck and Roger Haskin. GPFS: A shared-disk file system for large computing clusters. In *Proc. of the First Conference on File and Storage Technologies (FAST)*, pages 231–244, January 2002.
- [21] D. Sundaram-Stukel and M.K. Vernon. Predictive Analysis of a Wavefront Application Using LogGP. In *PPoPP '99: Proceedings of the seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 141–150, 1999.
- [22] W.Gropp and E.L. Lusk. Reproducible measurements of MPI performance characteristics. In *PVM/MPI*, pages 11–18, 1999.

Restart in Competitive Environments

Katinka Wolter* Philipp Reinecke†

Abstract

Retransmissions counteract message loss in many protocols for reliable message exchange. The approach has been generalised to restarting slow jobs with the intention of reducing completion times. A common concern with respect to the effectiveness of the restart method is whether it will be beneficial if used by every one and every task and whether it will harm those using a longer timeout or not restarting at all.

We present two models that indicate that adaptive restart mechanisms such as the restart algorithm in [vMW06] will avoid harmful overload by the efficient selection of a sufficiently large timeout value. This short paper aims at shedding light on the competitive restart scenario. We do so by proposing two different queueing models that represent the situation of multiple users competing for resource usage. The simulations of our two different models strongly indicate that in a scenario of high load with a highly variable job completion time distribution restart is especially suited to prevent the system from collapsing.

1 Introduction

Restart mechanisms are commonly applied in reliable protocols such as the TCP or the WSRM (Web Services Reliable Messaging) [KR01, BMT05] to ensure message transmission in the presence of message loss. Recently, they have also been studied as a means to reduce job completion times. When the completion time distribution is characterised by high variance, restart of slow jobs may yield a lower completion time on the next trial, thereby reducing the overall completion time of the job [RvMW04, RvMW06a, RvMW06b, vMW06]. This application of restart is motivated by an experience familiar to most users of the WWW: When loading a web page takes too long, clicking the ‘Reload’ button in many cases instantaneously reveals the desired page.

*Humboldt-Universität Berlin, Institut für Informatik, Unter den Linden 6, 10099 Berlin, Germany wolter@informatik.hu-berlin.de

†Humboldt-Universität Berlin, Institut für Informatik, Unter den Linden 6, 10099 Berlin, Germany preineck@informatik.hu-berlin.de

Intuitively, completion times (e.g. the transmission times of messages sent by a WSRM implementation) depend, among other factors, on the load on the system. In a congested network, message transmissions cannot be expected to be fast. Furthermore, with high load one expects correlation between subsequent completion times. In this case, restarting slow jobs will not reduce completion times. Even worse, restart generates more jobs and may thus increase completion times further.

Obviously, the efficiency and effects of restart depend on the restart frequency. This frequency is determined by the timeout after which a job is restarted (e.g. a SOAP message is resent by the WSRM). While restart timeouts may be set beforehand, commonly the timeout is adjusted based on observations of previous completion times (cf. the classification in [RvMW06b]).

In this work we aim to investigate the evolution of the restart timeout under increasing load. We focus on the algorithm proposed in [vMW04], which determines a restart timeout that minimises the expected completion time, based on observations of the completion time distribution. The assumption is that transmission times will increase with heavier load and hence the retransmission (or restart) timeout will increase as well. In consequence, in a heavily loaded system less restarts will be triggered by the restart algorithm, avoiding congestion.

To investigate this situation we set up two different queueing models. The first is a simple single server model. This model includes queueing and jobs are restarted if their response time exceeds a given timeout. As the response time of a job consists of the waiting time as well as the service time of a job it is not a simple random variable behaving according to some distribution. Service of each job, however, is straightforward.

The second model avoids queueing and represents the behaviour of the network as load-dependent service rate. Each job is served immediately, but at varying service rate.

Both models are implemented in a simulation in Mathematica and compared in their response time and queue length (including the number of jobs in service). Simulation analysis strongly supports our presumption that restart is of particular aid in managing highly loaded systems.

2 Basic Restart Model

The basic restart model is very simple [vMW06]. A task is started, and when it has not completed at a threshold time, it is retried. The task is assumed to complete according to some probability distribution, and it is assumed that each retry terminates the previous attempt (synchronous, abortable task execution in [RvMW06b]). One question to be asked is: In order to minimise the completion time, what is the best time to restart? If long transmission times are caused by congestion it might be wise not to restart too soon. Reducing the load in

a network may be the most efficient way to speed up transmission times. As a consequence, different optimisations are possible and the restart time can be selected in different ways. In earlier work we have studied static as well as adaptive timeout selection algorithms [RvMW06a] and their effect on the transmission time.

One practical instance of this model is restart as employed in a Web Services Reliable Messaging implementation: The WSRM source sends SOAP messages to the WSRM destination using a SOAP transport. The most common SOAP transport is the HTTP. The WSRM destination acknowledges received SOAP messages to the WSRM source. If the WSRM source does not receive an acknowledgement for a message before a timeout elapses, it resends this message. In this scenario, retransmissions may be triggered by message loss and by delayed transmissions, e.g. due to stalling TCP connections. The WSRM source must determine a timeout that ensures fast transmission of messages. However, if the timeout is set too low, overload may result.

It should be noted that this example presents a simplified view on WSRM. In general, the WSRM source cannot abort previous transmissions; furthermore, WSRM implementations usually send several messages in parallel instead of sequentially.

3 Single Server Queueing Model

In the simple single server queueing model we assume that jobs can arrive from different sources at different rate. Since the arrival process to the queue is a renewal process we can compute the arrival rate as the sum of the rates of the single arrivals and simply consider one arrival stream. For the time being the restart timeout is identical for all arrival streams and is based on the overall response times.

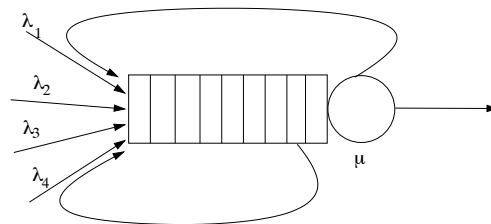


Figure 1: Single server queueing model

The single server queueing system is shown in Figure 1. Jobs arrive at rate $\lambda = \sum_i \lambda_i$ to the queue. Each job draws a randomly distributed service time. While waiting in line, the timeout value for the job is decreased. It can happen that

even before the job enters service the timeout expires and the job is reinserted into the queue using a newly drawn random service time.

For exponentially distributed interarrival and service times this situation corresponds to a reordering of the queue, which should not have any effect on the mean response time.

The model has the disadvantage that job response times consist of the service time and the waiting time. With a restart the service time is newly drawn from a distribution, but the waiting time depends on the length of the queue. Even a shorter service time upon the next trial may in some cases not coincide with a shorter job response time. It has the advantage that in the simplest case, when interarrival as well as service times are exponentially distributed, it corresponds to an M/M/1 queueing system. This M/M/1 queue could easily be solved analytically, would not restarts complicate matters.

Other authors use several queues [MM04], while we in this model assume jobs to return to the same queue using the same server. In our solution algorithm we simulate a queueing system with exponentially distributed interarrival time, one server, restarting jobs and an arbitrary service time distribution.

4 Degrading-Rate Infinite Server Queueing Model

We arrive at our second model from a system perspective: There is one server with service rate μ and m agents, each generating jobs with an arrival rate λ . Every job arriving at the server is immediately served, i.e. there is no queueing involved. In contrast to the well-known $M|M|\infty$ and $M|M|m$ queues, the service rate of the server is shared among all simultaneously served jobs, i.e.

$$\mu_i = \frac{\mu}{i}, \text{ for } i = 1, 2, \dots, m \text{ jobs.}$$

Each job generated by an agent k is assigned a timeout τ_k . Jobs that do not finish within the timeout are removed from the system and restarted. We model restart by drawing a new service time.

This model addresses the competitive aspects involved when restart is applied by several parties accessing the same resource. In particular, it promises to offer the following benefits over the single-server queueing model:

- (1) It allows us to study the effects of incomplete knowledge on the part of the agents, since each agent bases its restart timeout only on observations of its own completion times.
- (2) It models the effects of high load directly, by sharing the available capacity between all agents, and reducing the service rate accordingly.
- (3) It is open to extensions modeling more complex job characteristics. For instance, one may consider jobs that comprise a (randomly drawn) ‘thinking

period', during which the job does not occupy server resources (i.e. $\mu = k = \mu/(k-w_k)$, where $0 \leq w_k \leq k$ is the number of 'thinking' jobs). Such periods may be observed with e.g. TCP connections that stall during connection setup [KR01, RvMW06a, RvMW06b, RW08].

On the other hand, the model is much more complex. In particular, it may not lend itself easily to elegant solution techniques. Furthermore, we note that this model is not fully developed yet and may perhaps be reduced to a system of m parallel queues with load-dependent servers.

5 Analysis and Comparison of the Models

We have implemented both models in Mathematica simulations. In this section we show results from both implementations.

In both models we use the Lognormal distribution for the work requirement of a job, or the service time distribution. We parameterise the Lognormal distribution such that the squared coefficient of variation equals 12, i.e. is quite large. The parameters μ and σ of the Lognormal distribution are set to $\mu = 0.3$ and $\sigma = 2$. This implies that the expected service time $E[S] = 10$.

With exponentially distributed interarrival times the first model corresponds to a modified M/G/1 queue. Jobs arrive at rate $\lambda = 0.08$. The utilisation of this queue, ignoring the restarts, then is $\rho = E[S] \cdot \lambda = 0.8$. Within the mission time of 10000 time units the queue could process 698 jobs while using 288 restarts. The mean job completion time equals 3.22. Of interest is the evolution of the queue length as well as the value of the timeout τ . Both are shown in Figure 3

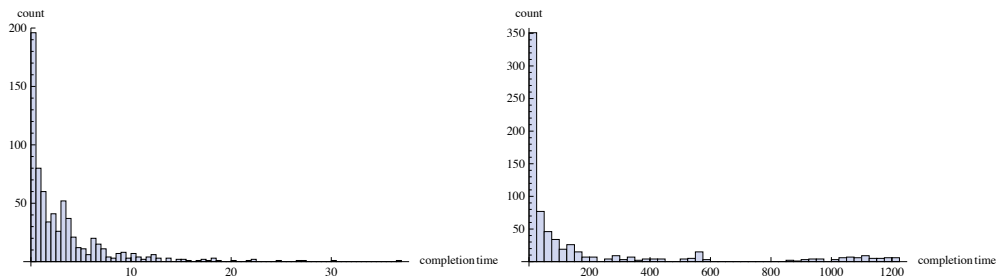


Figure 2: Histogram of completion times with (left) and without (right) restart and $\lambda = 0.08$

Without using restart the queue could process slightly more (713) jobs, but the mean response time was 169.93. This is due to some large outliers as shown in the histogram on the right hand side in Figure 2. In the given scenario with

relatively low load one clearly benefits from using restart. The queue length when using restart is much shorter and the mean completion time is much less. The restart timeout τ mostly stays below 30, which means that jobs are aborted after relatively short time. On the average almost every second job is restarted once and apparently the probability of sampling a short service time after few restarts is quite high.

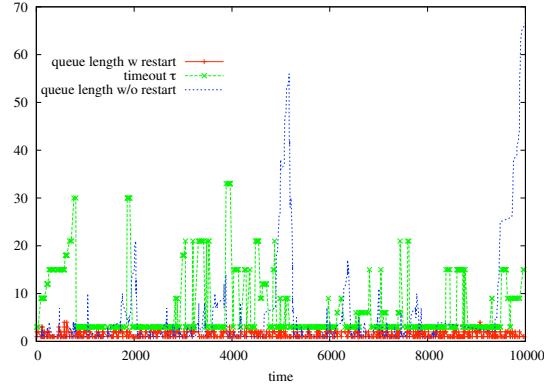


Figure 3: Queue length and timeout value for $\lambda = 0.08$

To evaluate the impact of higher load the arrival rate of jobs was increased to $\lambda = 0.095$, leading to a utilisation of $\rho = 0.95$ when ignoring restarts.

Without restarts 726 jobs are processed with a mean completion time of 211.26. The histogram looks similar to the one in Figure 2 on the right with slightly more frequent large values. It is omitted here.

Using restart the performance indicators of the queue are almost unchanged. The mean response time of jobs is 3.09191 and the queue has processed 878 jobs with 320 restarts. It should be noticed that these numbers are based on single simulation runs and are therefore within the normal limits of randomness. The graphs are omitted here as they are similar in structure to the ones shown above.

In this model increasing the load has an impact on completion times when not using restart, but with restart the system seems not to suffer.

The second model we ran for 10000 time units. In each run a constant number of agents was used, while increasing the number of agents in consecutive evaluations. Each agent draws a work requirement for its jobs from the same distribution. The agents process their work in a processor sharing fashion. After completion of a job, the agent pauses randomly a short time interval (or none) and then draws and processes the next job. We compare results with

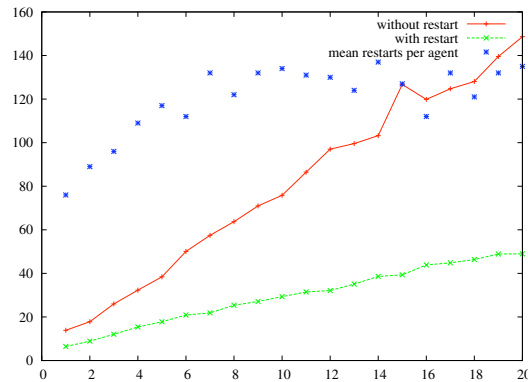


Figure 4: Mean job completion times in the second model

and without restart, only using our QEST algorithm [vMW06] to determine the restart interval length.

The results are shown in Figure 4. The solid lines show the mean completion time of the jobs when using the number of agents as indicated on the x-axis, competing for the server. Clearly, the mean completion time per job increases when more agents share the server. But, interestingly, this increase is much less when agents are allowed to restart their job in case it takes too long. The dotted curve shows the average number of restarts an agent performs during its working time of 10000 time units. Initially, the number of restarts an agent performs increases, saturating at around 130 restarts.

Figure 5 shows the number of jobs that can be processed by the system in total when using the number of agents as given on the x-axis and with or without using restart. Using restarts a fixed number of agents is able to complete up to a factor 3 as many jobs. The number of restarts in Figure 4 can now be interpreted as follows. If 20 agents are using a server they process in total roughly 7500 jobs, so each agent can complete on the average 375 jobs. While executing those 375 jobs the agent uses roughly 130 restarts. On the average almost one out of three jobs experiences a restart.

If there are only 10 agents in the system, each agent applies the same number of restarts (roughly 130), but each agent processes on the average 650 jobs, resulting in a total of 6500 jobs, so the agent will restart only one in 5 jobs.

As the system becomes more congested restart is applied more frequently, resulting in an only mildly increasing completion time with an increased processing capacity in terms of the number of completed jobs. Concluding from this analysis restart should be favoured more strongly the more congested a system becomes.

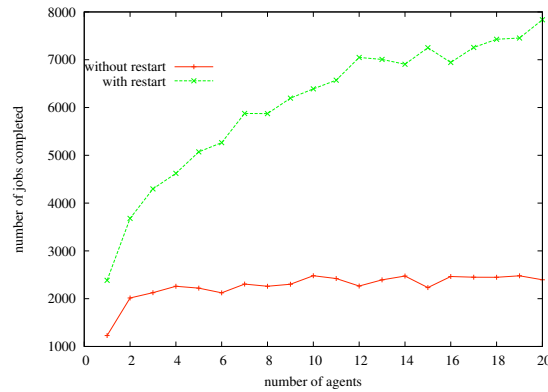


Figure 5: Number of jobs completed within the mission time

Comparing the two models is not straight forward. The simple queueing model can be seen as an implementation of the second model with only one agent. Then job completion times with restart are in a similar order of magnitude.

6 Conclusion

In this short paper we have presented two different queueing models designed to analyse the effects of restart in a heavily loaded environment. We want to study the effects of restarted jobs on job completion time as well as on the system load.

The first model is a single server model, corresponding in the simplest case to an M/M/1 queue. The model is complicated by allowing jobs to drop out and reenter the queue using a new randomly drawn service time. In this model the queue length can be interpreted as system load, which can be studied together with the response time. However, the response time as such is not a random variable. Instead it depends on the waiting time as well as the service time. The processing speed of the server is constant at all times and the response time of a job depends on its service time as well as the waiting time in the queue. This model is not able to represent slow processing due to congestion. Heavy load will only show in long waiting times.

We therefore propose a second model, where all jobs are served simultaneously in a round-robin, or processor sharing, fashion, leading to a load-dependent service rate. The more jobs are in the system the slower the server becomes. Here again jobs can drop out and return to the queue, immediately entering service again.

The results from our models are promising, but this paper presents ongoing work that leaves some questions unanswered.

In particular the common special cases of the two models with similar results still need to be identified.

References

- [BIMT05] BEA Systems, IBM, Microsoft Corporation Inc, and TIBCO Software Inc. *Web Services Reliable Messaging Protocol (WS-ReliableMessaging)*, February 2005.
- [KR01] B. Krishnamurthy and J. Rexford. *Web Protocols and Practice*. Addison Wesley, 2001.
- [MM04] S. Martin and I. Mitrani. Dynamic Routing Between Two Queues with Unreliable Servers. *International Journal of Simulation*, 5(5):38–48, 2004. Simulation Society.
- [RvMW04] P. Reinecke, A. van Moorsel, and K. Wolter. A Measurement Study of the Interplay between Application Level Restart and Transport Protocol. In *Proc. International Service Availability Symposium (ISAS)*, number 3335 in Lecture Notes in Computer Science, Munich, Germany, May 2004. Springer.
- [RvMW06a] P. Reinecke, A. van Moorsel, and K. Wolter. Experimental Analysis of the Correlation of HTTP GET Invocations. In *Proc. European Performance Engineering Workshop (EPEW)*, number 4054 in Lecture Notes in Computer Science, pages 226–237, Budapest, Hungary, June 2006. Springer.
- [RvMW06b] P. Reinecke, A. P. A. van Moorsel, and K. Wolter. The Fast and the Fair: A Fault-Injection-Driven Comparison of Restart Oracles for Reliable Web Services. In *Proc. 3rd International Conference on the Quantitative Evaluation of Systems (QEST) 2006*, Riverside, CA, USA, September 2006. IEEE.
- [RW08] P. Reinecke and K. Wolter. Phase-Type Approximations for Message Transmission Times in Web Services Reliable Messaging. In *Proc. SPEC International Performance Engineering Workshop (SIPEW)*, volume 5119 of *LNCS*, Darmstadt, Germany, June 2008. Springer-Verlag. to appear.
- [vMW04] A. van Moorsel and K. Wolter. Analysis and Algorithms for Restart. In *Proc. 1st International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 195–204, Twente, The Netherlands, September 2004. Best paper award.
- [vMW06] A. P. A. van Moorsel and K. Wolter. Analysis of restart mechanisms in software systems. *IEEE Transactions on Software Engineering*, 32(8), August 2006.

A PEPA model of a threshold policy sleeping server

Nigel Thomas *

Abstract

In this paper a model of a server which may undergo periods of sleep when idle is presented using the Markovian process algebra PEPA. The distributions used in this model are assumed to be of a phase type. This model can be applied to the study of systems where the conservation of energy is of particular concern, e.g. mobile devices or server farms. This gives rise to a trade-off between response time and power consumption which is explored numerically.

1 Introduction

Queueing models involving various forms of server vacation have been studied for many years. The case where a server can enter a *sleep* mode when the queue is empty has recently taken on more relevance with growing interest in power conservation. There are two broad areas where conserving power is of interest. The first is mobile communications, where devices have a limited battery life and prolonging the life of the battery is the principal performance measure of interest. The second case is large scale systems, such as server farms or computational grids, where there are many machines experiencing varying service demands. In this situation it is considered advantageous to power down a subset of the available servers at times of low demand in order to reduce costs, whilst still maintaining an acceptable level of service quality.

The model presented here is related to a class of model sometimes referred to as *N*-policy queues, first studied by Yadin and Noar [11]. A server enters a sleep period when the queue becomes empty and remains sleeping until there have been a sufficient number of arrivals (i.e. when the queue has *N* waiting jobs). The server then wakes up and serves the jobs until the queue is empty. The model in this paper is different from earlier work on *N*-policy queues in that it is assumed that turning the server off and on takes time (and consumes power), whereas in most previous studies (with the exception of [7]) it is assumed that these transitions are instantaneous and cost free. A further implication for the model of this assumption is that additional requests may arrive into the queue whilst the server is powering up or powering down. A very small number of other studies have also considered queues to have finite capacity, most notably in relation to this paper is Wang *et al* [10].

In this paper a sleeping server model is specified using the Markovian process algebra PEPA. A number of assumptions are made that enable this model to be

*School of Computing Science, Newcastle University, UK *Nigel.Thomas@ncl.ac.uk*

analysed using the PEPA Workbench [3]: the queue is assumed to be finite, the service, power up and power down time distributions are assumed to be of phase type and the arrival distribution is assumed to be Markov modulated Poisson to give bursty arrivals.

In the following section a brief overview is given of PEPA and how it is used to model phase type distributions. A more detailed description of the model and its formal specification is then presented, followed by some numerical results. Finally some conclusions are presented along with some possible directions for future work.

2 PEPA

A formal presentation of PEPA is given in [5], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the *type* of the activity, α , and the *rate* of the associated negative exponential distribution, r . This rate may be any positive real number, or given as *unspecified* using the symbol \top . The syntax for describing components is given as:

$$P ::= (\alpha, r).P \mid P + Q \mid P/L \mid P \bowtie_L Q \mid A$$

The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P . The component $P + Q$ behaves either like P or like Q , the resultant behaviour being given by the first activity to complete.

The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ .

Concurrent components can be synchronised, $P \bowtie_L Q$, such that activities in the *cooperation set* L involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to activities of that type. The parallel combinator \parallel is used as shorthand to denote synchronisation with no shared activities, i.e. $P \parallel Q \equiv P \bowtie_{\emptyset} Q$. $A \stackrel{\text{def}}{=} P$ gives the constant A the behaviour of the component P .

In this paper we consider only models which are cyclic, that is, every derivative of components P and Q are reachable in the model description $P \bowtie_L Q$. Necessary conditions for a cyclic model may be defined on the component and model definitions without recourse to the entire state space of the model

2.1 Phase-type distributions

The exponential distribution is not always the most interesting to employ when considering models of computer networks. In addition it is not necessarily the most realistic for practical applications, particularly arrival processes. Although PEPA cannot be used to model general distributions, it can be used to specify phase type distributions. Phase type distributions are distributions constructed by combining multiple exponential random variables. These can be used to

approximate most general distributions and approximations can be constructed using tools such as *EMpht* [1].

The Erlang distribution is a commonly used example of a phase type distribution which consists of an exponential distribution repeated k times. The PDF for the Erlang distribution is as follows:

$$f(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$$

In PEPA this is generally modelled as a ticking clock:

$$\begin{aligned} \text{Clock}_i &\stackrel{\text{def}}{=} (\text{tick}, t). \text{Clock}_{i-1} \quad , \quad 1 < i \leq k \\ \text{Clock}_1 &\stackrel{\text{def}}{=} (\text{event}, t). \text{Clock}_k \end{aligned}$$

Where $t = k\lambda$. The Erlang distribution is generally used to approximate deterministic events; the greater the value of k (i.e. the more ticks) the more deterministic the Erlang distribution becomes. However, it should also be noted that the larger the value of k , the more states there will be in the underlying CTMC. Hence, although we may wish to have 40 or 50 ticks to generate a nearly deterministic process, in practise a typical value of k is in the range [5, 10]. When studying network protocols the Erlang distribution is very useful for modelling timeouts.

Another important phase type distribution is the hyper-exponential, or H_k , distribution, which is a random choice between k exponential distributions. The most commonly used hyper-exponential is the H_2 -distribution, which has three parameters, α , μ_1 and μ_2 and the following cumulative distribution function.

$$F_{H_2} = 1 - \alpha e^{-\mu_1 t} - (1 - \alpha) e^{-\mu_2 t} \quad , \quad t \geq 0.$$

In PEPA this branching cannot be modelled explicitly except by introducing a new pair of actions over a choice operator. However, in practise branching may be represented implicitly at the preceding action. Thus if a request arrives into an empty queue and gets a hyper-exponential service, it could be represented thus:

$$\begin{aligned} \text{Queue}_0 &\stackrel{\text{def}}{=} (\text{arrival}, \alpha a). \text{Queue}_{1a} + (\text{arrival}, (1 - \alpha) a). \text{Queue}_{1b} \\ \text{Queue}_{1a} &\stackrel{\text{def}}{=} (\text{service}, \mu_1). \text{Queue}_0 \\ \text{Queue}_{1b} &\stackrel{\text{def}}{=} (\text{service}, \mu_2). \text{Queue}_0 \end{aligned}$$

In this representation the probabilistic branch is made by the choice of one of two *arrival* actions. The arrivals themselves will occur at the rate λ , i.e. the sum of the two branches under the race condition. Alternatively, as in the model in this paper, a service component can be constructed as follows:

$$\begin{aligned} \text{Server} &\stackrel{\text{def}}{=} (\text{service}, \alpha \mu_1). \text{Server} + (\text{service}, (1 - \alpha) \mu_1). \text{Server}' \\ \text{Server}' &\stackrel{\text{def}}{=} (\text{service}, \alpha \mu_2). \text{Server} + (\text{service}, (1 - \alpha) \mu_2). \text{Server}' \end{aligned}$$

It is important to note that in this representation the *Server* component will always perform a *service* action at rate μ_1 first before having the opportunity

to branch. Subsequent *service* actions will occur at either rate according to the branching probability α . This behaviour would not affect the overall steady state solution (since the start state is irrelevant in steady state when the model is irreducible), however it should be noted that inconsistent results may arise when transient or passage time analysis is naively applied to such a model.

An important feature of the hyper-exponential distribution is that it has a greater variance than an exponential distribution of the same mean (as long as $\mu_1 \neq \mu_2$ obviously). This is in contrast to the Erlang distribution (for example); thus by choosing between Erlang, exponential and hyper-exponential it is possible to consider a wide range of behaviours.

An important class of distributions for network modelling concerns ‘bursty’ arrival processes. The most common way of modelling this is to use Markov modulated Poisson process (MMPP). In PEPA this is simple to model as follows:

$$\begin{aligned} Arrivals_{off} &\stackrel{def}{=} (turnOn, \gamma).Arrivals_{on} \\ Arrivals_{on} &\stackrel{def}{=} (arrival, \lambda).Arrivals_{on} + (turnOff, \beta).Arrivals_{off} \end{aligned}$$

There are many other phase type distributions that can be used and good approximations to most general distributions can be made. It is worth noting however that the more phases considered, the greater the impact on the size of the state space, which can be a limiting factor for some forms of analysis.

3 The Model

The system is modelled as a simple single server finite queue. Requests arrive at the queue according to a two state Markov modulated Poisson process. In the ‘on’ state, arrivals occur at a rate λ and in the ‘off’ state no arrivals occur. Transitions occur between these states according to negative exponential random variables; from ‘on’ to ‘off’ at rate β , and from ‘off’ to ‘on’ at rate γ .

If the queue is not full, requests will be accepted and processed on a first come first served basis. If the server is awake, then requests are processed according to a two-phase hyper-exponential distribution (H_2), with parameters α , μ_1 and μ_2 . The specification of the queue follows the usual state-based approach for models of this nature, see [8] for example.

If the queue is empty then the server may enter a sleep period. It takes an Erlang-K distributed period to power down the server (*poweroff* with mean $1/\xi$). During this period requests may continue to arrive and be queued, but the server is committed to shutdown. Once the server has shutdown, it enters the *sleep* mode, where requests continue to arrive and are queued. Once there are N requests in the queue, the server begins to power up (again, Erlang-K distributed, with mean $1/\eta$). Once again, during this process more requests may arrive and be queued. Note that the power up process can only begin after the power down process has completed. Therefore, in the unlikely event that N or more arrivals occur during powering down, the server will not begin to power up immediately, i.e. there is no equivalent of the *powerup* action in *Shutdown_i* for $i \geq N$.

$$Sleep_i \stackrel{def}{=} (arrival, \top).Sleep_{i+1} \quad 0 \leq i < N$$

$$\begin{aligned}
Sleep_i &\stackrel{def}{=} (arrival, \top).Sleep_{i+1} + (powerup, \top).Queue_i \\
&\quad + (tock, \top).Sleep_i \quad N \leq i \leq n \\
Sleep_n &\stackrel{def}{=} (powerup, \eta).Queue_n + (tock, \top).Sleep_n \\
Shutdown_i &\stackrel{def}{=} (poweroff, \top).Sleep_i + (tick, \top).Shutdown_i \\
&\quad + (arrival, a).Shutdown_{i+1} \quad 0 \leq i < n \\
Shutdown_n &\stackrel{def}{=} (poweroff, \top).Sleep_n + (tick, \top).Shutdown_n \\
Queue_1 &\stackrel{def}{=} (service, \top).Shutdown_0 + (arrival, \top).Queue_2 \\
Queue_i &\stackrel{def}{=} (service, \top).Queue_{i-1} + (arrival, \top).Queue_{i+1} \quad 1 < i < n \\
Queue_n &\stackrel{def}{=} (service, \top).Queue_{n-1} \\
Arrive_{on} &\stackrel{def}{=} (arrival, \lambda).Arrive_{on} + (off, \beta).Arrive_{off} \\
Arrive_{off} &\stackrel{def}{=} (on, \gamma).Arrive_{on} \\
Server_0 &\stackrel{def}{=} (service, \alpha\mu_1).Server_0 + (service, (1-\alpha)\mu_1).Server_1 \\
Server_1 &\stackrel{def}{=} (service, \alpha\mu_2).Server_0 + (service, (1-\alpha)\mu_2).Server_1 \\
PowerOff_1 &\stackrel{def}{=} (poweroff, \xi/K).PowerOff_K \\
PowerOff_i &\stackrel{def}{=} (tick, \xi/K).PowerOff_{i-1} \quad 1 < i \leq K \\
PowerUp_1 &\stackrel{def}{=} (powerup, \eta/K).PowerUp_K \\
PowerUp_i &\stackrel{def}{=} (tock, \eta/K).PowerUp_{i-1} \quad 1 < i \leq K
\end{aligned}$$

$$(Arrive_{arrival} \boxtimes Sleep_0 \boxtimes_{service} Server_0) \boxtimes_{\mathcal{L}} (PowerOff_K || PowerUp_K)$$

Where $\mathcal{L} = \{poweroff, powerup, tick, tock\}$.

The model forms a continuous time Markov chain with $4(M + T + K(2M + 2 - T))$ states.

4 Numerical results

This model is now evaluated numerically using the PEPA Workbench [3]. In all experiments the phases of the Erlang distributed clocks was taken to be 6. The maximum queue size is limited at 30, hence this model has 794 states when $N = 1$.

The key element of this form of sleeping server is the optimal choice of the threshold value. If the threshold is too large then jobs will experience excessive waiting time and increase the probability that the queue will become full and jobs lost. In general, if the threshold is too small then the server will power up, serve the waiting jobs and power down on a regular basis, consuming resources (power and computation time) on these transitions. Ultimately the

optimum value will depend on the parameters chosen, most crucially the length and intensity of the arrival bursts. If the bursts are sufficiently intense then it will be beneficial to turn on the server as soon as the burst starts, i.e. the threshold will be very small. This is because the queue will be very unlikely to become empty during an arrival burst. Thus, the aims become to power up the server very soon after the start of the burst and power it down once all the jobs from the burst have been served.

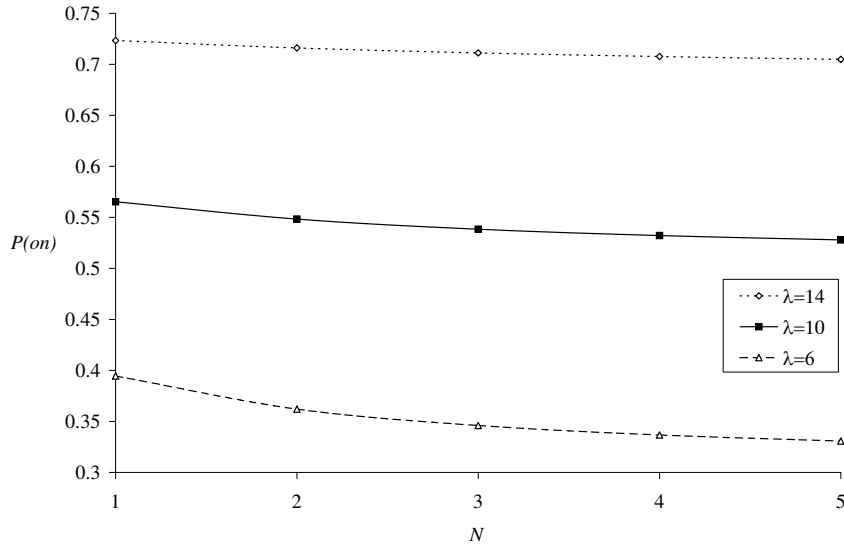


Figure 1: Proportion of time the server power is on, where the threshold value is varied for different arrival rates during bursts, $\mu_1 = \mu_2 = 10$, $\beta = \gamma = 1$, $\xi = \eta = 60$, $K = 6$.

Figures 1-3 show the performance of a single N -policy queue at different arrival rates.

As expected, in Figure 1, the power consumption is least when the threshold is largest and the response time is least when the threshold is 0. The profile of the power consumption plots changes considerably with load; the biggest gain in increasing the threshold is made when the load is lower. This is because at low load the queue is more likely to contain few jobs, whereas at higher load the queue is more likely to exceed the threshold during the burst, regardless of the threshold value. This implies that, for infrequent intense bursts ($\lambda \gg \bar{\mu}$), the threshold value is not, in fact, very important, and any threshold ($N \geq 1$) will achieve similar results.

The gradient of the response time plots in Figure 2 is not greatly influenced by load between $\lambda = 10$ and $\lambda = 14$, although obviously the response time is greater when the load is higher. However, the profile of $\lambda = 6$ is noticeably different and, surprisingly, at $N = 5$ the average response time for $\lambda = 6$ exceeds that of $\lambda = 10$. Taken in isolation this result would be quite perplexing, however looking at Figure 1 it is clear that the service time offered in these cases is quite different. This is because for any λ and N there is a small probability that a

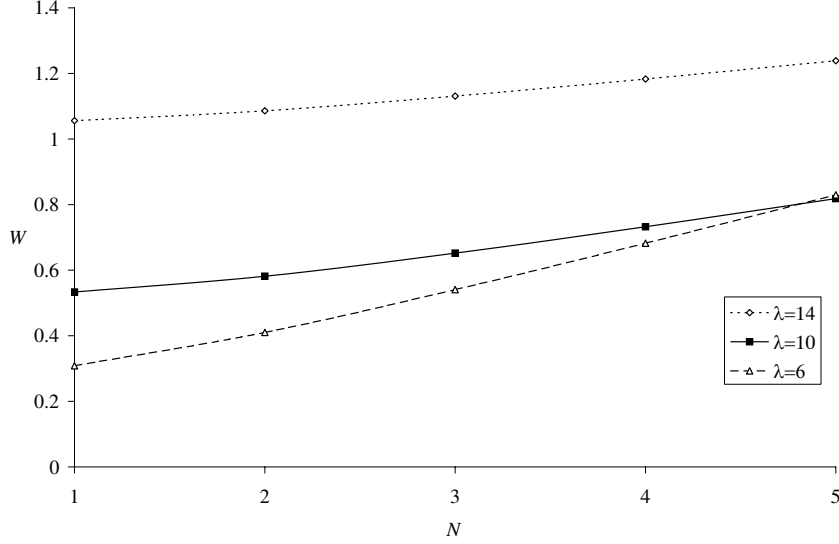


Figure 2: The average response time, where the threshold value is varied for different arrival rates during bursts, $\mu_1 = \mu_2 = 10$, $\beta = \gamma = 1$, $\xi = \eta = 60$, $K = 6$.

burst will not cause the queue to exceed the threshold. As N increases and λ decreases this probability becomes increasingly significant, to the point where it is likely that more than one burst is required to exceed the threshold. This means that, for low load and larger thresholds, an arrival is quite likely to be in the queue for some time before the server turns itself on and begins service.

In Figure 3 the rate at which jobs are lost is shown to not be greatly affected by the threshold value, even when $\lambda = 14$. This is because the server is fast enough that the queue rarely becomes full (causing loss). If the bursts were longer and more intense, then this may become more of an issue. The crucial observation here is that in this model there are no arrivals between bursts. If there was a low level of arrivals between intense bursts then the server would spend its time turning off and on at regular instants if the threshold was short, or allowing a few jobs to sit in the queue for a long time if the threshold was larger. This can easily be incorporated into our model by modifying the *Arrive_{on}* and *Arrive_{off}* behaviours.

$$\begin{aligned} \text{Arrive}_{on} &\stackrel{\text{def}}{=} (\text{arrival}, \lambda_{high}).\text{Arrive}_{on} + (\text{off}, \beta).\text{Arrive}_{off} \\ \text{Arrive}_{off} &\stackrel{\text{def}}{=} (\text{arrival}, \lambda_{low}).\text{Arrive}_{off} + (\text{on}, \gamma).\text{Arrive}_{on} \end{aligned}$$

The greater the difference between the two arrival rates, the more bursty the arrivals are; typically, $\lambda_{high} \gg \lambda_{low}$. It might be considered confusing to have arrivals during an *off* period, hence it would be clearer to rename these components *Arrive_{high}* and *Arrive_{low}*. We can now study the situation where

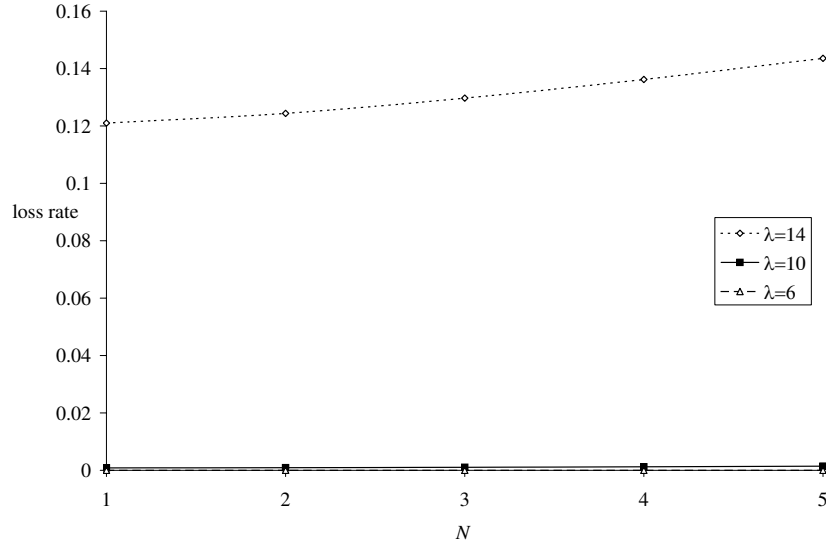


Figure 3: The rate of job loss, where the threshold value is varied for different arrival rates during bursts, $\mu_1 = \mu_2 = 10$, $\beta = \gamma = 1$, $\xi = \eta = 60$, $K = 6$.

there is a small, but persistent, trickle of jobs arriving between bursts, this is shown in Figures 4-6.

Figure 4 shows that the background traffic does indeed have an impact in the proportion of time that the server is on. It is interesting that with the background rate of $\lambda_{low} = 2$, this proportion is increased across all the values of N shown, whereas the intuition might be that the impact would be greater for $N = 1$ than $N = 5$. There is a clear difference between the server on times for different arrival rates, with the server being on for up to 80% more when $\lambda_{high} = 14$ than when $\lambda_{high} = 6$. However, there is a down side to this improved energy efficiency at low load, and this is shown in Figure 5. Here we see again (as in Figure 2) that jobs will reside in the system for longer whilst the server is off. However, as there is a small background load it is more likely that a burst will exceed the threshold, even when $\lambda = 6$. Further, if a burst just fails to reach the threshold, it will be exceeded soon after by further arrivals in the background traffic. Clearly, this means that the server is switching on and off more frequently and this is evident if we compare Figures 1 and 4.

The magnitude of the response time is also increased in Figure 5 from that in Figure 2. This is due to the overall increase in the average arrival rate (as $\lambda_{low} = 2$ instead of 0). Finally, Figure 6 shows that the loss rate is still not greatly impacted by this low level of background traffic. However, there is a small loss rate when load is higher and this does increase slightly as the threshold value increases (and the server is consequently turned on less).

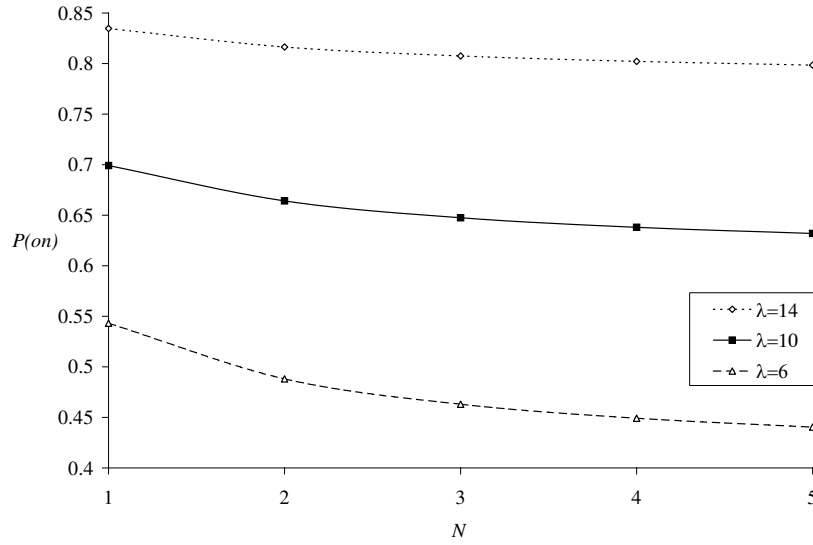


Figure 4: Proportion of time the server power is on, where the threshold value is varied for different arrival rates during bursts, $\mu_1 = \mu_2 = 10$, $\beta = \gamma = 1$, $\xi = \eta = 60$, $K = 6$, $\lambda_{low} = 1$.

5 Conclusions and Further Work

In this paper some initial results have been presented for an N policy queue. This is part of an ongoing investigation into the trade-off between performance and power consumption. Some of these results are slightly counter intuitive. In particular we show that a system with a low load of bursty traffic can have a slower response time than a similar system with slightly higher load. Clearly, the trade-off between power and performance as demonstrated here is subtle and complex and worthy of much further exploration.

References

- [1] S. Asmussen, O. Nerman and M. Olsson, Fitting phase-type distributions via the em algorithm, *Scandinavian Journal of Statistics*, 23, pp 419-441, 1996.
- [2] M. Bernardo, L. Donatiello and R. Gorrieri, Describing Queueing Systems with MPA, Technical Report UBLCS-94-11, University of Bologna, May, 1994.
- [3] G. Clark and S. Gilmore and J. Hillston and N. Thomas, Experiences with the PEPA Performance Modelling Tools, IEE Proceedings - Software, pp. 11-19, **146**(1), 1999.

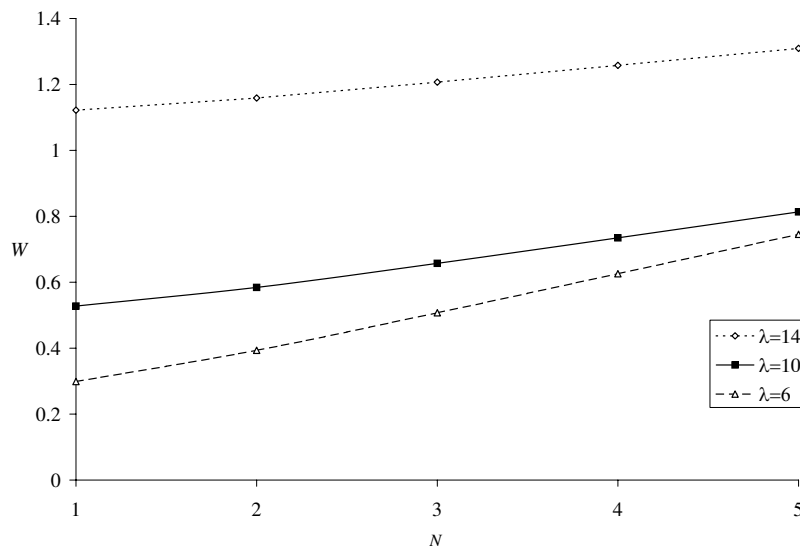


Figure 5: The average response time, where the threshold value is varied for different arrival rates during bursts, $\mu_1 = \mu_2 = 10$, $\beta = \gamma = 1$, $\xi = \eta = 60$, $K = 6$.

- [4] U. Herzog and V. Mertsiotakis, Stochastic Process Algebras Applied to Failure Modelling, in U. Herzog and M. Rettetbach (eds.), *Proc 2nd Workshop on Process Algebra and Performance Modelling*, 1994.
- [5] J. Hillston, *A Compositional Approach to Performance Modelling*, Distinguished Dissertations in Computer Science 12, Cambridge University Press, 1996.
- [6] J. Slegers, N. Thomas and I. Mitrani, Dynamic Server Allocation for Power and Performance, in: S. Kounev, I. Gorton and K. Sachs (eds.), *Proc SPEC International Performance Evaluation Workshop*, Springer Verlag, 2008.
- [7] M. Sobel, Optimal average-cost policy for a queue with start-up and shut-down costs, *Journal of Operations Research*, **17**, pp. 145-158, 1969.
- [8] N. Thomas and J. Hillston, Using Markovian Process Algebra to Specify Interactions in Queueing Systems, Technical Report LFCS-97-373, Department of Computer Science, University of Edinburgh, 1997.
- [9] N. Thomas, A simple model of a finite queue with a sleeping server, in: B. Haverkort and L. Cloth (eds.), *Proceedings 7th International Workshop on Performability Modeling of Computer and Communications Systems*, CTIT Workshop Proceedings, University of Twente, 2007.

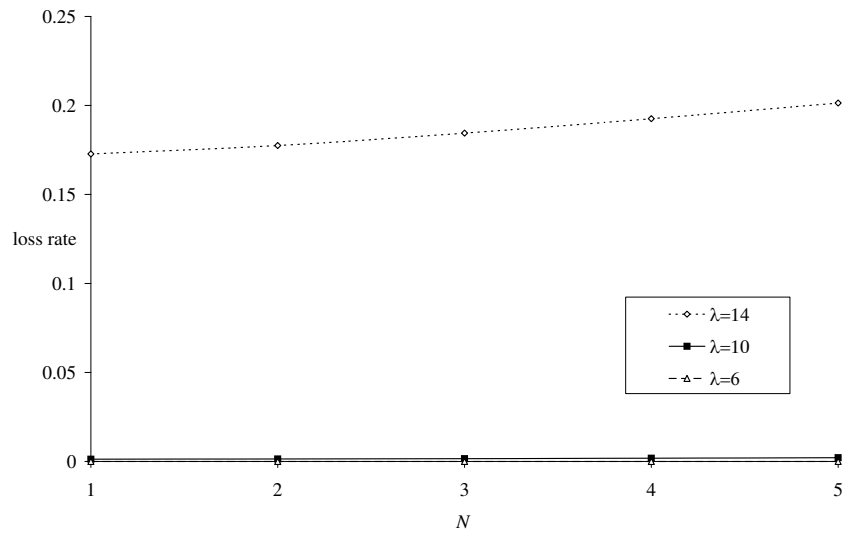


Figure 6: The rate of job loss, where the threshold value is varied for different arrival rates during bursts, $\mu_1 = \mu_2 = 10$, $\beta = \gamma = 1$, $\xi = \eta = 60$, $K = 6$.

- [10] K. Wang, K. Chang and B. Sivazlian, Optimal control of a removable and non-reliable server in an infinite and finite $M/H_2/1$ queueing system, *Applied Mathematical Modelling*, **23**, pp. 651-666, 1999.
- [11] M. Yadin and P. Noar, Queueing systems with a removable service station, *Operational Research Quarterly*, **4**, pp. 393-405, 1963.

A Middleware for Activating the Global Open Grid

Jeremy Cohen ^{*} Colin Richardson [†] John Darlington [‡]

Abstract

The continuing development of Computational Grids is being supported by extensive research in science and industry. The ability to share resources both within and across organisations provides opportunities for significantly increased resource utilisation. However, there are many issues with the sharing of resources, both in finding available resources for use and, particularly across organisations, with the transfer of funds between providers and consumers and the defining of Service Level Agreements (SLAs). In this paper we describe the MAGOG architecture – a Middleware for Activating the Global Open Grid – an alternative to standard middleware designs that is based on a peer-to-peer infrastructure. It is believed that MAGOG provides a highly scalable infrastructure to support the discovery of resources and agreement of usage terms for an open market in computational power.

1 Introduction

Computational Grids offer the potential for cross-organisational access to heterogeneous resources. The term Virtual Organisation (VO) is often used to refer to a logical grouping of entities from different physical organisations that work together in a Grid environment sharing resources. Support for VOs is provided through middleware, software that sits between users and the fabric of heterogeneous hardware resources that they wish to access. This middleware provides various services that allow users to discover available Grid resources and execute jobs on them.

Cross-organisational resource access within the context of a VO is helpful to users and can result in more efficient working environments once the hardware and software are configured. However, these Grid environments are generally the result of a collaboration between a group of individuals or organisations that are known to each other and where there is some level of trust between the parties that can be supported by technical security measures. Firewall rules may need to be altered to allow traffic to flow between organisations' systems and accounts or account mappings may be required so that users can execute jobs on the remote machines that are exposed through middleware installations.

^{*}Internet Centre, Department of Computing, Imperial College London, jhc02@doc.ic.ac.uk

[†]Internet Centre, Department of Computing, Imperial College London, clrich@doc.ic.ac.uk

[‡]Internet Centre, Department of Computing, Imperial College London, jd@doc.ic.ac.uk

However, as the development of Grids continues and, more recently, Utility Computing providers are appearing that offer remote access to resources in a pay-per-use, on-demand manner, the desire for easier access to remote resources is growing. The vision of Grid Computing is the emergence of a global Grid of resources owned by many different entities, accessible on-demand, with users paying for the processing time that they use. Some of the significant issues with reaching this vision are how resources are discovered, how their usage is negotiated and how payment is transferred between parties who are unknown to each other and untrusted by each other. In this paper we present a technical view of MAGOG – a Middleware for Activating the Global Open Grid – whose design was conceived by Colin Richardson in 2006 and set out in [6] and [7]. MAGOG is based on a peer-to-peer architecture and takes a radical approach to discovering remote resources and negotiating for access to them. It is based on three key concepts: the ‘catallaxy’ paradigm of economics, the ‘small-worlds’ principle and a double message-flooding algorithm. We describe the design of the middleware and look at possible techniques for implementing such a system on top of the existing Internet.

The rest of the paper is organised as follows. Section 2 provides an overview of the concepts underlying MAGOG, with the full architecture being detailed in Section 3. In Section 4 we look at implementation techniques and in Section 5 we discuss expected market behaviour of the middleware with conclusions and further work covered in Section 6.

2 Concepts

The MAGOG middleware is based on 3 key concepts; Catallaxy, the ‘small-worlds’ paradigm and use of a double message-flooding algorithm.

Catallaxy: The term Catallaxy was first used by economist Friedrich von Hayek to define the costless emergence of a stable state in a decentralised market within which all entities aim to maximise their utility. The use of this paradigm in the study of computational infrastructures has also been attempted within the CATNETS project [1] where it has been used in the provision of an efficient resource allocation and scheduling mechanism for Grid environments.

The ‘small-worlds’ paradigm: Stanley Milgram observed from his work [4] that there are, on average, six degrees of separation between any two individuals on earth. Taking the model of human networking and applying this to computing networks, it can be seen that the models are not dissimilar. At the edges of the network are entities that have a small number of connections to others, moving into the middle of the network, better connected nodes are reached that have knowledge of much larger numbers of nodes and how they can be reached. Applying this small-worlds paradigm to computer networks gives us the small-world network. From this model it can be seen that in a network with a structure that approximates a small-world network, any node with a connection to the outside world should be able to reach any other node within an average of six hops.

Double message-flooding: In networks, particularly within peer-to-peer environments, one of the approaches taken to get messages out to a number of nodes is to send a broadcast message. When these messages are designed to travel a long distance through a network, they are given a large TTL (time to live) value

Within the architecture there are one or more interlinked Grid Clearing House (GCH) entities. These entities are responsible for secure, accurate accounting of the agreements made in the environment and for handling legal issues such as SLA management and resolution of SLA-related disputes.

A payment service (the IPS entity in Figure 1) is responsible for handling payment between parties in a deal. Settlement occurs monthly, in accordance with standard commercial practice.

The process of striking a deal follows a standard bid and ask approach similar to that used in stock markets. The analogy of bees is used to illustrate the ‘swarms’ of data packets that are sent out containing the bid and ask messages and the various other packets sent during the agreement process.

An entity wishing to sell resource capacity sends out a swarm of AskBees. These go out across the network, traversing the various links available to them. An entity wishing to purchase resource capacity sends out a swarm of BidBees. As the Bees pass through MAGOG nodes, they are handled by the MAGOG stack (illustrated in Figure 2).

Each MAGOG node runs an implementation of the MAGOG stack. The lower two layers of the stack use protocols and utility packages specific to a given MAGOG deployment platform, the upper layers are platform independent and are the same in all deployments. The bottom two layers provide transport and security so that packets are secured and have a means to be transported over the underlying open network (the Internet).

The third level of the stack is where matchmaking takes place. Bids and asks are resolved at this layer. Incoming bid and ask packets are compared with each other and with other packets that have passed through the node recently. A copy of incoming bids and asks is stored for later comparison with subsequent incoming bids and asks, while a copy of each incoming bid or ask is forwarded back onto the network to move on to the next node in the network (providing its TTL has not reached zero).

When a bid and ask match, a deal is made. At this point, level 4 of the stack takes over. A deal results in the generation of a ChekBee, a packet that is securely transmitted to the payment service to ensure payment is remitted for the deal. In order for the deal to be sealed, a DealBee must reach the Grid Clearing House (GCH). The DealBee is created on receipt of a confirmation from the IPS that payment is possible. The check against the IPS does not result in any funds transfer. It is a credit check that ensures all entities involved in the deal are genuine and that the purchaser has the ability to pay the agreed sum. To denote satisfactory completion of the transaction after resource usage has taken place, both parties must send an EndBee to the GCH.

3.1 Service Level Agreements

Service Level Agreements are an extremely important aspect of the MAGOG environment. An entity wishing to obtain access to a resource bids for some required capacity and usage terms. An entity offering a resource broadcasts an ask stating the availability of the resource and usage terms. SLAs provide a contract that formalises the agreed usage terms when a deal is made. SLAs may cover a large number of properties but key issues are ensuring that consumers are provided with the performance and job priority that they have requested. If a user wants access to an execution resource of a given performance in 5 minutes

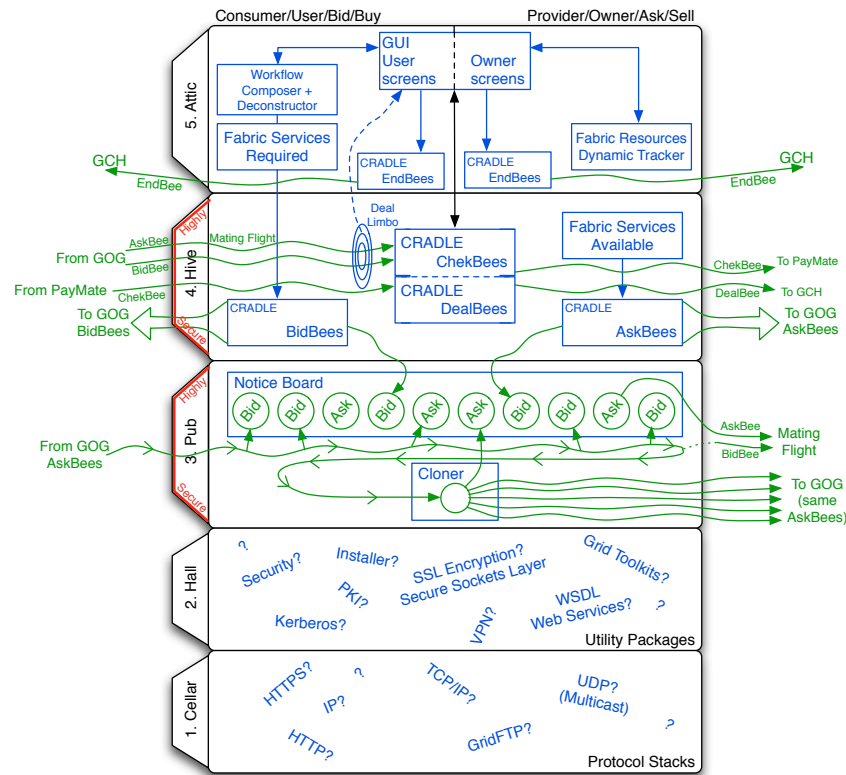


Figure 2: The MAGOG stack.

to carry out a job that will take 1 hour, obtaining access to a faster resource, but with a lower job priority, meaning that the job does not start for 12 hours is not satisfactory. Such requirements need to be encapsulated within an SLA.

There may be incentive for dishonest providers to offer unrealistic SLAs in order to gain business that they would otherwise lose to more reliable providers. In most cases, this can be detected as part of the verification carried out when a deal is made. A matching bid-ask pair move to the potential provider's resource(s) where identity verification and credit checking take place. Additional checks on reservation databases can be made to ensure that the claims made by an ask packet relate to genuinely available resource capacity. Nonetheless, SLA specification should be supported by a suitable means for dispute resolution in the event that an SLA is violated. If the complete process of deal making, SLA agreement and dispute resolution can be handled programmatically in all but the most complex cases, this allows for an environment where interaction between providers and consumers is optimised and should result in a larger number of deals taking place within the market.

The use of reputation mechanisms whereby parties in a deal can rate their level of satisfaction with the service or product provided can help to enhance the SLA process. Consumers rate the quality of various transaction properties

and this information can then be used to help other consumers select an entity to agree a deal with in future transactions. It should be noted that while reputations provide a method to assist with identifying rogue traders, they can also be used by business rivals to discredit each other by filing false information on a provider's Quality of Service (QoS) and such issues must be taken into account when using reputation information.

3.2 Payment

Payment requires a secure system that allows details of necessary funds transfers to be recorded within the MAGOG network. This is accomplished using the MAGOG Chek, Deal and EndBee messages. All these messages must be secure, encrypted messages. When a deal is made as the result of a bid and ask match, a ChekBee is sent to the payment service (IPS) to verify the identity of the parties involved and the credit rating of the consumer. This encrypted message is digitally signed by all parties involved in the transaction and the digital signatures can be used to verify, beyond reasonable doubt, the identities of the parties involved in the deal. If the identity and credit checking succeed, a DealBee is generated and sent to the Grid Clearing House providing a storable record of the agreement made between the provider and consumer. This message is again digitally signed by all parties involved in the transaction denoting acceptance of the agreed details of the deal and SLA that are encapsulated within the deal message. The agreed resource usage takes place and, once completed, an End message is sent from each party involved in the deal to the Grid Clearing House to confirm completion of the deal. A transfer of funds has not yet taken place. At some point in the future, the Clearing House carries out an account clearance process, sending details of all transactions that have taken place in the previous month. The payment service takes this information and clears funds between all entities as necessary in order to settle payment for all deals that have taken place.

4 Implementation Techniques

Implementation of the MAGOG framework is yet to begin, however various techniques have been considered for the development of the framework. Each node that takes part in the MAGOG environment will need to run an implementation of the MAGOG middleware stack. This is a small application that is described in more detail in Section 4.2. The packets of data that are transmitted between nodes need to be small, but also need to contain sufficient data to specify a node's requirements or offer – the bid or ask. The use of XML has been considered as a suitable way of describing this information in a platform independent manner that is easily processed. XML is somewhat verbose, however there are other ways to represent XML content for more efficient transmission and this may present a solution that makes XML viable. JSON (JavaScript Object Notation) [3] is a potential alternative that offers a more lightweight solution to transmission of data and could be suited to the representation of MAGOG data packets.

The data packets are generated at level 4 of the stack and pass down through the lower layers for transmission out onto the network. At level 2, the data is

prepared for transmission. This includes tasks such as message compression or altering XML content for more efficient transmission and the addition of security measures. While transport layer security may be used to secure data on the wire, MAGOG requires message level security for all packets due to the sensitivity of data that they may carry. This may be provided through security specific to a particular messaging model, for example if using Web Services, WS-Security [5] provides ideal message level security, or it may be provided through a standard encryption module within the MAGOG stack using an algorithm such as AES. Messaging is discussed in more detail in Section 4.1.

The need to ensure that messages are not tampered with means that the MAGOG stack digitally signs all bid and ask packets that are created in addition to payment and deal related packets. This ensures that a receiver of a packet can tell, beyond all reasonable doubt, which node generated the packet. Traceability can be further enhanced by adding an audit trail that involves every node that handles a packet attaching a further digital signature created using the handling entity's private key. This ensures that it is possible to reliably identify every node that has handled a packet but adds overhead to the system. Profiling of the signing process will be required to see how significant the overhead is and whether the process can be optimised to support deployment of the system involving potentially hundreds of thousands or even millions of nodes. Using standard X.509-based digital signatures may be acceptable but this requires that all MAGOG nodes are issued with a personal X.509 certificate that is signed by a trusted Certificate Authority. The potential use of certificates by all nodes needs to be investigated further before a decision is made on this element of the implementation.

Given that wide uptake of MAGOG is necessary to provide a liquid market with sufficient providers and consumers, the barriers to taking part in the system must be low. Installation of the MAGOG middleware stack must be a simple and efficient task that can be handled by relatively non-technical users. Highly reliable operation of the middleware is important to ensure that market participants are not deterred from using the platform once it is running.

4.1 Messaging

Messaging protocols are used to transmit information between MAGOG nodes. The messages transmitted between nodes represent bids or asks for resource capacity, or network configuration messages that may be broadcast around the network to optimise network organisation or bidding patterns. Message formats use a platform independent data representation such as XML so that the message content is transport agnostic.

Nodes may support multiple messaging formats to increase the flexibility for use of the MAGOG infrastructure in environments where different messaging formats are preferred. Figure 3 shows an example network structure where node B is able to send and receive messages using three different messaging protocols, JMS, RMI and SOAP over HTTP (Web Services). Node A transmits an ask onto the network to offer some resource capacity and Node F transmits a bid. The double message flooding architecture results in the bid and ask messages being evaluated at Node B and a deal being made. While support for several different messaging formats in a single node is likely to be uncommon, this example is used to illustrate the flexibility of the MAGOG architecture allow-

ing development of implementations of the MAGOG stack supporting different messaging environments.

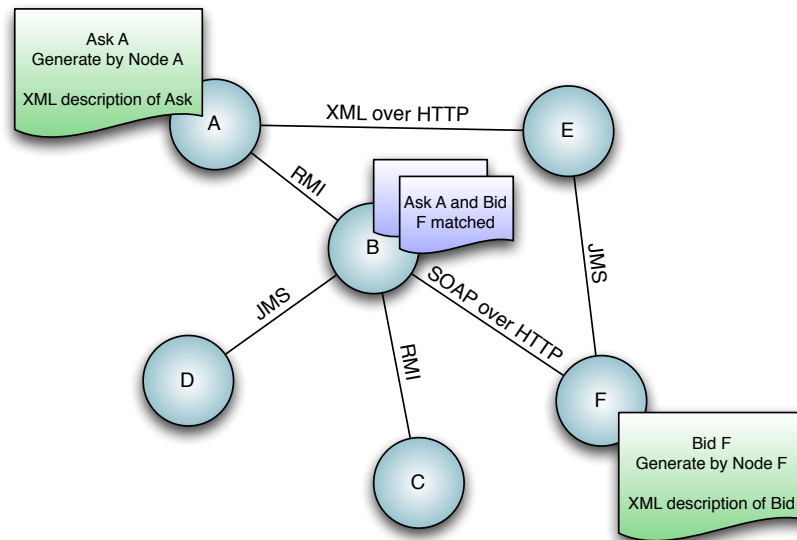


Figure 3: **Communication between nodes using different protocols.**

4.2 MAGOG Daemon

The MAGOG Daemon is the service that must run on a node that is to interact within the MAGOG environment. The daemon listens on one or more ports in order to accept messages in one or more supported messaging formats and provides an implementation of the MAGOG stack shown in Figure 2. The daemon stores incoming messages and then broadcasts them on to other known nodes within the network. This forwarding may be through direct unicast transmission to a list of a small number of known nodes, or using multicast to take advantage of the networking infrastructure to broadcast the messages to a greater number of MAGOG nodes listening on a given multicast address.

Received messages are stored in a local cache for resolution against other recently received messages. The resolution process involves evaluating each message in the cache against every other looking for potential deals. If a deal is made, the daemon is capable of generating credit checking and audit packets that are sent to the Payment Service (IPS) and Clearing House (GCH) respectively, using clients for these two services that are embedded in the daemon service. Figure 4 shows a schematic of the MAGOG daemon that sits on a resource between the local operating system and any Distributed Resource Manager or Reservation Service. When a deal is made for use of the local resource, either locally or at a remote location, the local MAGOG middleware instance is notified and communicates the details of the deal that will result in local resource access to the local DRM system or reservation manager to ensure

that the resource is reserved for the time slot of the deal. This notification can be initiated by the BidBee of a bid-ask pair that arrive at a resource shortly after a bid-ask match has been made for its use.

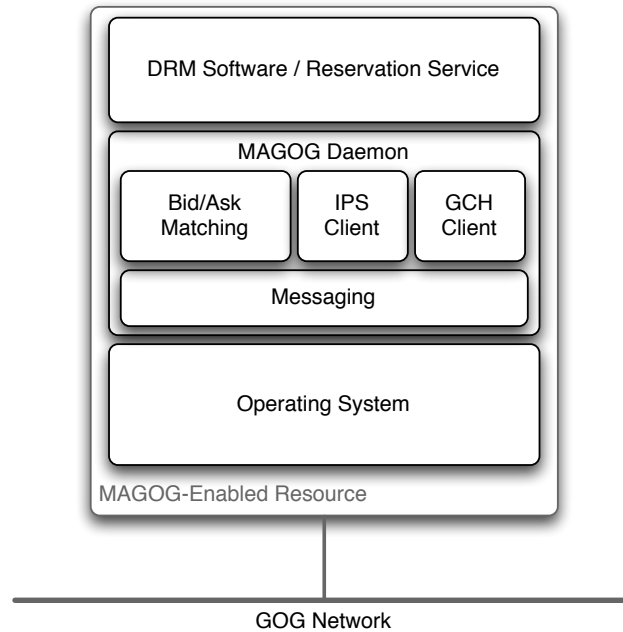


Figure 4: The MAGOG daemon on a MAGOG-enabled resource.

5 Market Behaviour

In [1], Torsten Eymann et al. state that catallaxy is derived from the Greek ‘katallatein’, meaning ‘to barter’ and ‘to join a community’. The catallaxy is a state of coordinated individual actions, brought about by the bartering, communicating and relationship-building activities of economic agents, leading them to achieve an unplanned community goal. This ‘spontaneous order’ of stable relative prices emerging from numerous transactions in a competitive market is Friedrich von Hayek’s 1930s take on Adam Smith’s 1780s ‘invisible hand’. Prices are not only rates of exchange between goods, but also an efficient mechanism for communicating information between localities and communities. In a complex, uncertain environment, economic agents are never able to predict the precise consequences of their actions, and it is this existential ignorance that actually makes the price system work.

The spontaneous order can never be designed by a perfectly informed planner who simply ‘gets the prices right’, because the pattern of pricing actually evolves as a result of lack of knowledge. Most of this knowledge is generated and used during the process of adjustment to equilibrium, which cannot occur in the alternative economic paradigm of “general equilibrium”, proposed by Leon Walras in the 1870s. This is because the Walrasian auctioneer forbids ‘false

trading' (i.e. deals struck at non-equilibrium prices) until open outcry bids and asks reveal the economy's equilibrium vector of relative prices. Yet false trading is the very process that drives the catallaxy to emerge: the market process is necessarily one of incessant trial-and-error, according to Hayek.

The operation of the MAGOG middleware needs to be tested to ensure that separate bid and ask prices coalesce into 'deal prices', which then acquire the stability predicted by the catallaxy paradigm. Some initial modelling has been carried out in [2]. The aims are to ensure that the bid and ask process operates successfully and that from a given state of supply and demand, an equilibrium is reached.

6 Conclusions and Further Work

We have presented a technical view of the MAGOG middleware that has been developed as a radical alternative to standard Grid middlewares. MAGOG is expected to offer significant scalability advantages over existing middleware platforms in addition to superior solutions for management of legal and payment issues within a market-based Grid environment.

Initial simulation work on the MAGOG platform is ongoing and we plan a more substantial stage of simulation followed by prototyping the platform and ultimately producing a full implementation of the system for deployment over a global Grid testbed such as the PlanetLab system.

References

- [1] T. Eymann et al. Catallaxy-based grid markets. *Multiagent Grid Systems*, 1(4):297–307, December 2005.
- [2] U. Harder and F. Martinez Ortuno. Simulation of a peer to peer market for grid computing. In *15th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA 2008)*, volume 5055 of *LNCS*, pages 234–248, Nicosia, Cyprus, June 2008.
- [3] JavaScript Object Notation (JSON). <http://www.json.org/>. [accessed 12 June 2008].
- [4] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.
- [5] A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker, editors. *Web Services Security: SOAP Message Security 1.1*. Oasis Open, February 2006. available at <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>. [accessed 12 June 2008].
- [6] C. Richardson. Growing the global open grid: Design brief and middleware architecture. Technical report, Internet Centre, Imperial College London, London, UK, 2007.
- [7] C. Richardson. Utilising hayek's catallaxy in a global open grid. In *10th Anniversary Conference of the Association for Heterodox Economics*, Cambridge, UK, July 2008. Anglia Ruskin University. (To appear).

The performance of locality-aware topologies for peer-to-peer live streaming

Richard G. Clegg^{*}, David Griffin[†], Raul Landa[‡],
Eleni Mykoniati[§] and Miguel Rio^{¶||}

Abstract

This paper is concerned with the effect of overlay network topology on the performance of live streaming peer-to-peer systems. The paper focuses on the evaluation of topologies which are aware of the delays experienced between different peers on the network. Metrics are defined which assess the topologies in terms of delay, bandwidth usage and resilience to peer drop-out. Several topology creation algorithms are tested and the metrics are measured in a simple simulation testbed. This gives an assessment of the type of gains which might be expected from locality awareness in peer-to-peer networks.

1 Introduction

This paper investigates the impact of the topology of overlay networks on performance metrics for peer-to-peer live streaming. An overlay network is a conceptual network of peers which exists on top of the standard Internet. Peers on the overlay network connect according to given rules to form a topology. There has been recent research interest in making overlay networks locality-aware for so that peers may more easily find “nearby” peers. In this paper we undertake a systematic evaluation of a number of alternative locality-aware topology construction methods (and some random methods for comparison).

The situation considered is that of a single node, known as the *peer-caster* wishing to distribute live streaming content through a peer-to-peer network. The peers in the network wish to download this content reliably and with a low delay between the peer-caster and themselves. The challenge of distributing live content is somewhat different to that of distributing recorded content on demand. A major difference is that delay is important to optimise (so that peers can view streams as “live” as possible) whereas throughput only needs to be large enough to view the stream (a peer cannot continue to download at faster than the rate the stream is broadcast).

A number of strategies might be considered for forming such topologies for live streaming. Minimising delay to the peer-caster might be one strategy. Connecting to close (in terms of delay) nodes might be a related strategy. Another aspect to consider is whether

^{*}richard@richardclegg.org

[†]dgriffin@ee.ucl.ac.uk

[‡]r.landa@ee.ucl.ac.uk

[§]e.mykoniati@ee.ucl.ac.uk

[¶]m.rio@ee.ucl.ac.uk

^{||}Department of Electrical Engineering, University College London, London

it is important to aggressively minimise delay or closeness by making as many connections as possible to the lowest delay/closest node or whether it might be preferable to have a range of connections. The topologies formed are tested against several metrics which attempt to assess whether the topology is good at reducing delay, resilient in the case of peers dropping out and whether it ensures that the bandwidth is used fairly.

1.1 Background and related work

Distributing content over an overlay network has been the subject of numerous studies in recent years. Most of this research has been concentrated on non-live content where the emphasis is on increasing throughput rather than reducing delay. In early approaches like SpreadIt [4], a multicast tree is built by centralised logic running at the data source. Upon the arrival of a new node, the source is contacted to appoint an unsaturated node to be the parent of the new node. When the *smart-placement* policy is in effect, the parent node is also selected to be close to the new node, where proximity is inferred with traceroute messages. More recently, Bos [7] proposed a method which constructs a data distribution tree containing the *Euclidean Minimum Spanning Tree*, where the distance in the Euclidean space represents the network delay. A subset of stable and high capacity nodes are elected to become *super peers*. Super peers are interconnected to form a *Yao graph*, a structure which contains the Euclidean Minimum Spanning Tree. Normal peers attach directly to the closest super peer. The source routed multicast tree is built over the super peers topology based on the compass routing protocol.

The departure of a node in single distribution tree topologies results in complete loss of connectivity for all the nodes in the underlying subtree. To overcome this problem, several studies investigate streaming the data over a forest of multicast trees, each of which carries only part of the stream. Coopnet [8] is a forest-based streaming approach, where the authors identify a tradeoff between efficiency in terms of locality and path diversity required for resilience to node departures. Upon addition of a new node, the source returns a significantly large set of candidate parent nodes to ensure diversity. As an optimisation the candidate parent nodes are selected, similarly to SpreadIt, so that they are *nearby* the newly added node.

Techniques for constructing trees typically assume global knowledge and at least one interaction with the source. Alternatively, overlay topologies can be constructed with local knowledge, where the connections are determined by each node and the data flow may take many alternative and potentially overlapping paths. In [9] a technique for clustering nodes to bins based on their locality is proposed. As a case study of this technique, the *BinShort-Long* overlay construction method is presented, where each node connects to $k/2$ randomly selected nodes from within its cluster (bin) and $k/2$ random nodes from anywhere in the system. A similar technique is proposed in [1] as an improvement for the BitTorrent protocol. The clustering here is done primarily to distinguish between nodes located in the same ISP, and nodes in different ISPs. Out of the total BitTorrent peers discovered by a new peer through the local tracker, all but k are selected to be local peers, with typical values 35 for total peers and 1 for the k external peers. This is done to reduce the traffic over the inter-domain links while still maintaining enough connections with external peers to receive the data. Finally, in [10] the authors formulate the *Minimum Delay Mesh problem* and prove that it is NP-hard. They propose a heuristic for constructing a shallow (low number of hops) and locality-aware (low delay at each hop) overlay topology. In order to minimise the number of hops, nodes with higher capacity need to be connected closer to the source.

The selection of the nodes to establish connections with, is done after calculating the *power* of each node, as a function of the node's locality and bandwidth availability.

2 Simulation methodology

In order to make the simulation of the overlay tractable it is necessary to abstract away the network itself and simulate only the overlay. The simulation described here makes as few assumptions as possible. It is assumed that each node has a fixed delay to every other node in the overlay (as described in the next section). It is also assumed that each node has a sufficient download bandwidth to obtain the entire stream and upload bandwidth to deliver a fixed proportion (which may be more than unity) of the stream.

2.1 Node distributions

Synthetic coordinate systems associate a coordinate with each peer in an overlay network, in such a way that the distance between the coordinates is a good estimate of some network property measured between the peers, predominantly *round trip time* (RTT). This can be achieved efficiently by using a limited set of end-to-end measurements to extrapolate those distances between nodes that were not explicitly measured. Thus, synthetic coordinate systems use a limited set of measurements to model the structural properties of the Internet, and then use this model to predict end-to-end properties (such as RTT) between arbitrary peers.

The first step in the operation of a network coordinates system is generating a *distance graph*, where links between peers represent distance measurements. This distance graph is then *embedded* onto a space that integrates some of the structural properties of the Internet. Examples of these include a standard Euclidean space [2], a Euclidean space augmented with a purely additive coordinate [3] or a hyperbolic space [11]. The embedding process can be viewed as an error minimization procedure where nodes are positioned in the space in such a way that the cumulative difference between the measurements and the embedded distances is minimized. Once this embedding has been done, and to the extent that the embedding space faithfully recovers the structure of the Internet for the measure in question, geodesic distances over this space are good predictors of the actual distances over the Internet [6]. This space will be referred to as *delay space*.

In the case of the simple simulation used in this paper, a standard two-dimension Euclidean delay space is used. Let N be the number of nodes in the system excluding the peercaster. The $N + 1$ nodes, numbered from 0 (the peercaster) to N are distributed over the two-dimensional Euclidean space. Each node has a co-ordinate (x_i, y_i) and the delay from node i to node j is obtained using the standard Euclidean distance from (x_i, y_i) to (x_j, y_j) .

The question then becomes how to distribute the nodes on the delay space. For the purposes of this paper we use three generation methods to create random node distributions. In reality, nodes in an overlay network will cluster to some degree, for example, nodes in the real Internet are more prevalent in some areas of the world than others (clusters in large cities, particularly large cities with high levels of Internet usage). In the case of an overlay network based upon nodes wishing to download particular streaming content, the distribution will be further complicated by whether the content is of regional, national or global interest as well as what language the broadcast is in.

For this reason the simulation here is tested against different assumptions about how nodes might be randomly situated in delay space.

Flat node distribution \mathcal{N}_F : In this distribution the nodes are flatly distributed in a square delay space. For each node i , x_i and y_i are chosen randomly from a flat distribution in the interval $(-D, D)$. In the simulations given here $D = 0.25$ seconds (so the maximum delay between any two nodes is $\sqrt{2}/2$ secs).

Tightly clustered node distribution \mathcal{N}_T : This distribution simulates a situation where nodes are grouped into tight clusters. The following procedure is followed until sufficient nodes have been generated.

1. Coordinates position X, Y is chosen with a flat distribution where X and Y are chosen from the interval $(-D, D)$.
2. The position (X, Y) is modified by a small random perturbation (d_X, d_Y) where d_X and d_Y are chosen with a flat distribution in the interval $(-d, d)$.
3. Coordinate (X, Y) is recorded.
4. With probability p go to step 1, otherwise go to step 2.

In this distribution $D = 0.25$, $d = 0.005$ and $p = 0.01$.

Loosely clustered node distribution \mathcal{N}_L : This distribution is identical to the previous one but the clusters are more diffuse and contain fewer nodes $D = 0.25$, $d = 0.05$ and $p = 0.01$.

In each of the last two cases, after the distribution is created, the node order is randomised. Node order is important for local topology schemes (see section 3.2). This reordering prevents nodes being created in a convenient “by cluster” order with nodes locally close being created together.

2.2 Modelling assumptions

For simplicity it is assumed that each node attempts to download a stream as M separate and equally sized *substreams* – note, however, that this could also be thought of as simply an abstraction of, say, a chunk-based swarming system with M partners from whom equal amounts are downloaded. Assume that each node has capacity to download all M substreams and that nodes have upload capacities to upload only a limited number of substreams.

Each node has associated with it an upload capacity u_i which is the number of substreams it can support (for the purposes of bandwidth calculation each substream is considered to have a bandwidth of 1Mb/s – although the precise unit is unimportant and of the metrics described, only the bandwidth variance is affected by this). Note that it must be assumed that $u_0 \geq M$ (in order that all M substreams can be uploaded from the peer-caster itself) and also for the system to scale it is important that $\bar{u}_i \geq M$ (the average peer has sufficient capacity to upload all M substreams). An implicit assumption is that system bottlenecks are only at the peers in the network. This may not always be the case in reality (for example several peers who belong to the same ISP may share access network capacity in the underlying network).

Nodes will then attempt to connect to at most M other peers in order to download the complete stream (nodes can download all M streams from a single partner node). A node i will accept at most u_i connections and request up to M connections. The complete set of connections will be referred to as a *topology* on the overlay network. This will be described in the next section.

For this paper u_i will be chosen from a random distribution. In addition u_0 will be fixed since it has such an important role in the network. The values used are $M = 4$ and u_i is chosen with equal probability from the set $\{1, 5, 10, 16\}$ – in this simulation no nodes are complete free-riders although some nodes can only produce 1/4 of a complete stream. The mean value of u_i is 8 so the system easily has capacity for every node to download the stream. As previously stated u_0 is a critical parameter in the system so $u_0 = 16$ for all simulations – the peercaster is always assumed to have a reasonable amount of bandwidth. This is to prevent the simulation results being greatly dependent on this single random selection (a simulation where $u_0 = 1$ might get very different results from one with $u_0 = 16$ even if all else was the same).

3 Topology generation and assessment

A topology on the network is a directed graph which may have more than one edge from node i to node j (an edge represents node i sending a single substream to node j). Let I_i be the number of incoming connections to node i and O_i be the number of outgoing connections to node i . A *valid topology* on the network is one where $I_i = M$ for $i \in 1, \dots, N$, $I_0 = 0$ (every node apart from the peercaster is downloading a complete stream) and $O_i \leq u_i$ for all i (every node is within its upload bandwidth limit). In addition there must exist a walk from node 0 to every node i in the network (every node has a path to download from the peercaster).

3.1 Metrics

Because each node has M independent connections, variants on more usual network metrics are used here. For example, it is not simply the shortest path from a node to the peercaster to the node which is of interest but the path length along all paths.

The metrics listed in this session have been created with several considerations in mind. A “good” topology should have all or most of the following properties.

- Low delay to end nodes – this translates to nodes being able to view streams with good “liveness”.
- High resilience to churn – a peer-to-peer network is, by its nature, highly dynamic. The loss of any single node should not greatly affect the network.
- Diversity of paths – related to the above, an individual peer would want a diverse set of connections so that the loss of a single intermediate node will not affect every stream it is downloading.
- Equitable spread of bandwidth – the upload bandwidth requirements in carrying the stream is spread across all nodes rather than falling unduly on a small number of nodes.

Let $D_i(j)$ be the shortest path delay from node i to the peercaster assuming that the first “hop” is node i ’s j th connection. Let $P_i(j)$ be the set of nodes which connect node i to the peercaster for its j th connection (not including i and the peercaster itself). If i connects directly to the peercaster for its j th connection then $P_i(j) = \emptyset$. Note that in pathological cases this shortest path may go back through the node i itself.

Let V_i be the vulnerability of node i to the removal of a single node. This is the number of streams connecting node i to the peercaster which could potentially be disrupted by

the removal of a single node (not including the peercaster). It is zero if and only if every node is directly connected to the peercaster. It is M if the shortest paths $D_i(k)$ go through one node (not including the peercaster) for all k .

Let S_i be the vulnerability of the system to the removal of node i . It is, in a sense, the dual of V_i . It is the total number of streams $D_j(k)$ (where $j \neq i$) which could be broken if node i were removed from the system.

Single figure metrics (the following metrics produce a single number).

1. Mean minimum delay $\mathbf{D}_{\min} = \sum_{i=1}^N \min_j D_i(j)/N$ – this is the mean of the minimum delay from a given node to the peercaster.
2. Mean maximum delay $\mathbf{D}_{\max} = \sum_{i=1}^N \max_j D_i(j)/N$ – this is the mean of the maximum shortest path from a given node to the peercaster.
3. Maximum system vulnerability $\mathbf{S} = \max_i S_i/MN$ – this is the proportion of routes which could potentially be damaged by the removal of a single node. It will be one if there is a single node (apart from the peercaster) which can disrupt every transmission path and zero if there are no nodes which can damage paths (only possible if every node connects directly to the peercaster). This measure is similar to finding the node with maximum Betweenness-Centrality [5]. It is a measure of a worst case vulnerability to churn in the network.
4. Mean node vulnerability $\mathbf{V} = \sum_{i=1}^N (V_i)/NM$ – this is mean of V_i over all nodes. It is a measure of how vulnerable the average node is to churn. The value is one if every node i can have all M streams broken along the shortest path by the removal of a single node not including i or the peercaster. The value is zero if and only if every node connects directly to the peercaster.
5. Bandwidth variance $\mathcal{B}_v = \text{var}(O_i)$ (for $i > 0$) – this attempts to assess whether the system load is split evenly between all nodes. If this is low then all nodes are using similar quantities of bandwidth but if it is high then some nodes are using large amounts of bandwidth and some little. Nodes with zero upload capacity are ignored by this metric.

3.2 Topology generation methods

The topology generation methods are split into two types: global and local fixed.

Global methods begin with all peer nodes present in the system. Connections can be chosen (at least potentially) with regard to every other node present in the network. Such methods are unrealistic in a large-scale real peer-to-peer system but may provide insight into the performance levels that can be expected from a topology. This can be thought of as a complete information godlike view of the system and its connections.

Local fixed methods have peers appearing on the network in sequence, peercaster first. Each peer chooses all M connections before the next peer joins the network. (The first peer present must connect M times with the peercaster). Such methods are practical for real peer-to-peer networks (although they may involve more information than could be easily obtained from a real peer-to-peer network).

An obvious extension to this would be local reevaluating methods in which nodes appear in a fixed order and initially get their connections using local methods but may subsequently change connections as new, better, nodes appear. These topologies will be studied in further work.

3.3 Global topology generation methods

Random global topology \mathcal{TG}_R – this can be thought of as the base “worst” case for global topology generation. This first algorithm is described in detail despite its simplicity so that the reader can fully understand the implementation. Subsequent topology algorithms will be described fully but in a less verbose manner. Let S be the set of nodes which do not yet have M substreams $S := \{i \in 1, \dots, N\}$. Let C be the set containing only the peercaster (node 0) $C := \{0\}$.

1. Randomly pick nodes i from C and j from S .
2. Make an outgoing connection from i to j . Set $I_j := I_j + 1$ and $O_i := O_i + 1$.
3. If $j \notin C$ then $C := C \cup \{j\}$.
4. If $O_i = u_i$ then $C := C \setminus \{i\}$.
5. If $I_j = M$ then $S := S \setminus \{j\}$.
6. If $C = \emptyset$ then an invalid topology has been reached. Begin the algorithm afresh.
7. If $S = \emptyset$ then all nodes have been connected. Otherwise go to step 1.

Closest first global topology \mathcal{TG}_C – This algorithm attempts to minimise delays by connecting “close” nodes first. Let C be the set of nodes which are “connected” (can trace a route from the peercaster) and have capacity to spare (initially $C = \{0\}$). Each node i which does not yet have $I_i = M$ maintains a record of the node in the set $C \setminus \{i\}$ to which it has the smallest delay. The node n makes a connection to its closest node j . The node n is added to C if it is not already there. The node j is removed from C if its capacity is filled. The process continues until all nodes are connected with $I_i = M$.

Small world \mathcal{TG}_S – This topology has $M - 1$ connections picked according to the rules in \mathcal{TG}_C and one node picked randomly according to the rules in \mathcal{TG}_R . The idea being to form a small world type network with mainly local connections but one global connection.

3.4 Local fixed topology generation methods

Local random \mathcal{TLF}_R – In this algorithm, the nodes appear in order and connect to random nodes which have spare capacity. As it appears a node picks one node from the set of nodes with spare capacity at random and connect to it. This process is repeated until the node has M connections.

Local closest first \mathcal{TLF}_{C1} – In this algorithm, the nodes appear in order and each node, as it appears, makes connections to the closest node to it (in terms of delay) until either all M connections are made or that node runs out of upload capacity (in which case the next closest node is used and so on).

Local closest with diversity \mathcal{TLF}_{C2} – This topology is the same as \mathcal{TLF}_{C1} but the nodes attempt to connect to M different other nodes if possible, if no such nodes are available then the nodes connect to the closest node with which they only have one connection and so on. (For example the first node to appear must by necessity make all M connections to the peercaster itself).

Local minimum delay first \mathcal{TLF}_{D1} – This topology is formed by each node connecting to that node with capacity to spare from which it can get the lowest shortest path delay (in the same way as \mathcal{TLF}_{C1} did).

Local minimum delay with diversity \mathcal{TLF}_{D2} – This topology is the same as \mathcal{TLF}_{D1} but with the modification that the node attempts to connect to M different nodes if possible.

Local small world \mathcal{TLF}_S – This topology has $M - 1$ nodes picked according to \mathcal{TLF}_{C2} and one node picked randomly according to \mathcal{TLF}_R .

3.5 Criticisms of the metrics and topologies

While the selected metrics are intended to be proxies for the properties declared at the start of this section, it is important to remember that they do not have a simple and direct relationship with those properties. For example, the delay based metrics are intended as a proxy for the the delay from the peercaster to the end user. However, the actual delay experienced by the user will not directly relate to the metrics specified here. In a substreaming system for example, this would depend upon exactly which nodes that particular substream travelled through. This itself might be subject to optimisation separately from topology. It would also depend upon exactly where in the stream the particular peer chose to upload from. The delay would also depend on choices made by the peer since there would be a trade off between delay and reliability in such a system (a peer may decide to allow a time buffer to allow for future jitter in downloading a stream).

Because the focus of this paper is on the effects of topology rather than on the effects of packet scheduling and buffering strategies it was decided to use simpler simulations and metrics which are only proxies for the ideal measurement.

The topologies considered are only a small subset of the possible topology construction methods. A moment's consideration will come up with many more possibilities, however, the number of results presented in this paper is already large. Future work will investigate different topologies.

Of the topologies considered, an obvious criticism is that they all have complete and accurate knowledge of the system. This approach is taken because the aim of this work is to develop an optimal strategy for real peers to implement. In the case of the local topologies this is the system “to date” and in the case of the global topologies this is the entire universe of peers. No “churn” is accounted for, nodes only enter. Both topologies have potential problems with the possibility of connecting up unrealistic or impossible networks. In the case of the global topologies, topologies can be generated with a minimum cut less than the stream bandwidth. This problem will be addressed in subsequent work. In the case of local topologies it is possible that the nodes could appear in an order which made connecting the network impossible. For example, with $M = 4$, if nodes with $u_0 = 4$, $u_1 = 1$ appeared then node 2 could not find four upload connections. This problem is addressed by ensuring that the node order is such that valid local topologies are always possible.

4 Results

Figure 1 shows the effects of the node distribution algorithms. The scale is delay in milliseconds.

It is useful to show an example of topology creation. A simple situation with ten nodes is shown with the following upload capacities (beginning with the peercaster, node 0) (16, 1, 5, 5, 10, 10, 10, 16, 1, 5).

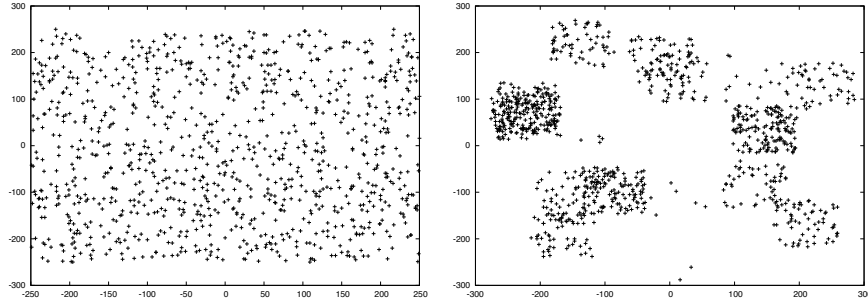


Figure 1: Flat \mathcal{N}_F (left) and loosely clustered \mathcal{N}_L (right) node distributions of 1000 nodes.

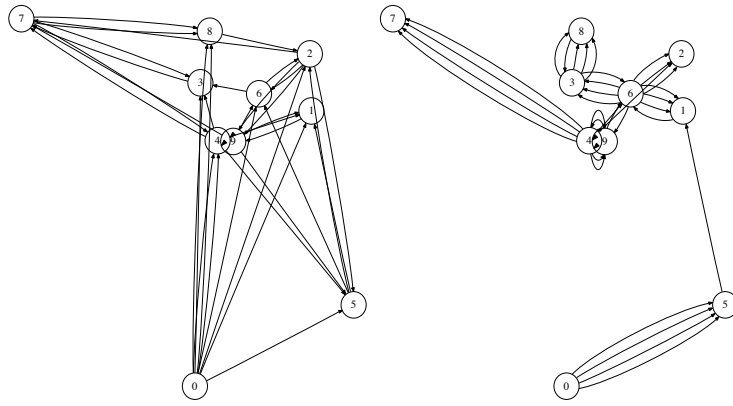


Figure 2: Ten nodes connected using the topologies global random \mathcal{TG}_R (left) and global closest \mathcal{TG}_C (right).

Figure 2 shows two of the global topologies, node 0 is the peercaster. The arrows indicate the direction in which data is transmitted. As might be expected the global closest topology \mathcal{TG}_C has many node pairs with a number of links in common. For example, all the upload links to node 5 come from the same node (the peercaster). This is because the first connection made is from node 0 to node 5 and after this connection node 5 is still the closest node to the set of connected nodes. Node 1 can only supply one upload stream because of its bandwidth constraints. In this topology nodes 1 and 6 mutually upload (this is entirely possible in a live streaming system). It should also be noted that the \mathcal{TG}_C topology is unrealistic in this case. The single link from node 5 to node 1 is a bottleneck in the network. Topology \mathcal{TG}_R as might be expected has much more diversity in connections.

Figure 3 shows the two local topologies \mathcal{TLC}_1 and \mathcal{TLC}_2 . As can be seen, in \mathcal{TLC}_1 it is common for a pair of nodes to have several connections. Indeed this pattern is only broken because of nodes running out of upload capacity (for example node 1). In topology \mathcal{TLC}_2 close connections are still made but nodes choose diverse connections. Because of the nature of the local topologies, node 1 has to connect to

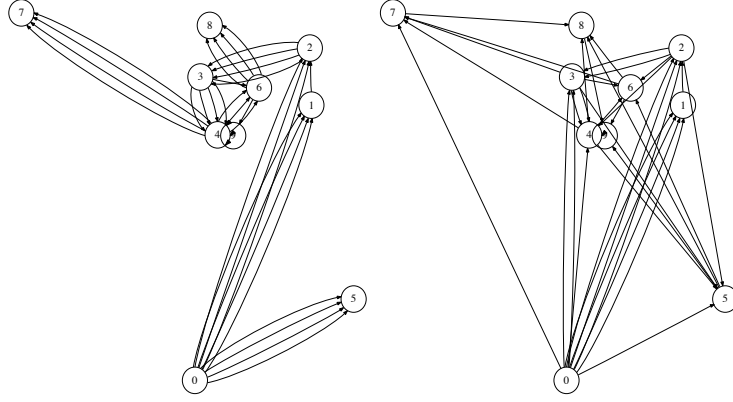


Figure 3: Ten nodes connected using the local closest topologies \mathcal{TLF}_{C1} (left) and \mathcal{TLF}_{C2} (right).

node 0 only (because at this point only node 1 and node 0 are in the system). Similarly node 2 can only connect to node 1 or node 0. The fact that each node has full upload bandwidth before other nodes can connect to it ensures that no unrealistic topologies (in the sense of the previous section) can be constructed using the local topology methods.

4.1 Experiments performed

For one of the node distributions \mathcal{N}_F , \mathcal{N}_L and \mathcal{N}_T 10,000 nodes are generated. Topologies are constructed from 100, 200, 500, 1,000, 2,000, 5,000 and 10,000 of these nodes (the nodes randomly chosen from the 10,000). To check the stability of each metric, each set of topology, node distribution and number of nodes is repeated ten times. The mean value of each of the metrics is calculated for the set of ten experiments and a 95% confidence interval is also obtained.

The majority of the results are presented with error bars representing the 95% confidence intervals. Note also that small offsets are introduced in the x-axis in order that the error bars do not overlap.

4.2 Delay metrics

Figure 4 shows \mathbf{D}_{\max} for the topologies on the flat node distribution \mathcal{N}_F . Of the global topologies, \mathcal{TG}_R is, as might be expected, the worst performing and scales badly with max delay increasing rapidly with the number of nodes in the system. \mathcal{TG}_C performs better but still does not scale well. \mathcal{TG}_S is the best performing and its scaling seems extremely good, indeed surprisingly so.

In the case of the local topologies, figure 4 shows that \mathcal{TLF}_{C2} has the lowest delay, with \mathcal{TLF}_S , \mathcal{TLF}_{C1} and \mathcal{TLF}_{D2} also performing well. Bad performers are \mathcal{TLF}_R (the random local topology) and, perhaps more surprisingly \mathcal{TLF}_{D1} . A possible explanation for the latter is that each node as it arrives attempts greedily to get its lowest delay to the peercaster. The earliest nodes to arrive connect directly to the peercaster. Once the peercaster bandwidth is exhausted a second wave of nodes has a high likelihood of connecting to any of those nodes close to the peercaster. Low delay nodes

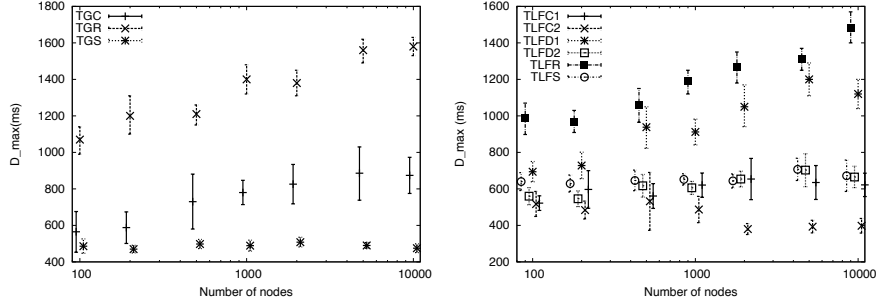


Figure 4: D_{\max} for global (left) and local topologies (right) on a flat node distribution \mathcal{N}_F .

would quickly find their upload bandwidth used up. The \mathcal{TLF}_{D1} topology reduces this effect somewhat by insisting that nodes upload from a diverse selection of other nodes.

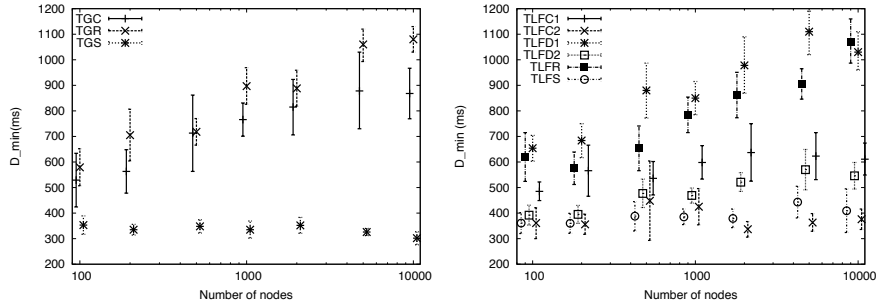


Figure 5: D_{\min} for global (left) and local topologies (right) on a flat node distribution \mathcal{N}_F .

Figure 5 shows the same experiment as figure 4 but uses the minimum delay metric D_{\min} . The graphs are essentially similar but \mathcal{TG}_R and \mathcal{TLF}_R perform better than previously. This is almost certainly a result of the fact that their random nature is likely to make the worst connection much worse than the best connection they experience. For the local topologies, \mathcal{TLF}_S , \mathcal{TLF}_{D2} and \mathcal{TLF}_{C2} are the best performing. Figure 6 shows the same results as figure 5 but with the tightly clustered node distribution \mathcal{N}_T . The differences between the graphs are not too great. The clustering seems to lower the minimum delay for most measures, perhaps because there are always some nodes extremely close to the peercaster. The only policy which appears significantly affected is \mathcal{TG}_C which performs consistently much better in the presence of clustering. Note especially that the topology has changed character from dramatically increased delay as the number of nodes increases to almost no increase in delay as the number of nodes increased. This is explored further in section 4.6.

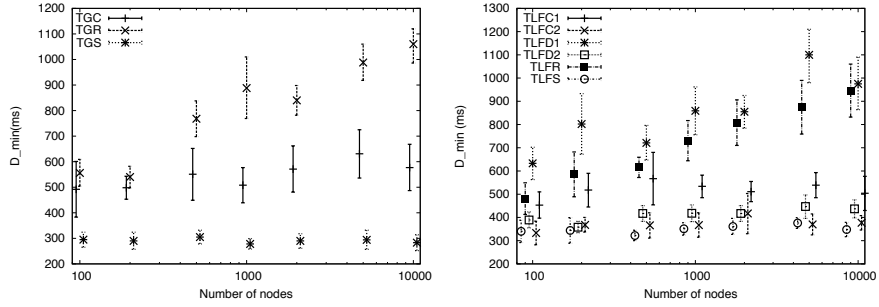


Figure 6: D_{\min} for global (left) and local topologies (right) on a tightly clustered node distribution \mathcal{N}_T .

4.3 Vulnerability metrics

Two vulnerability metrics were defined in section 3.1. The first \mathbf{S} attempts to assess the vulnerability of the system to the removal of a single node. The second \mathbf{V} attempts to measure the vulnerability of the average node to the removal of some other node from the system.

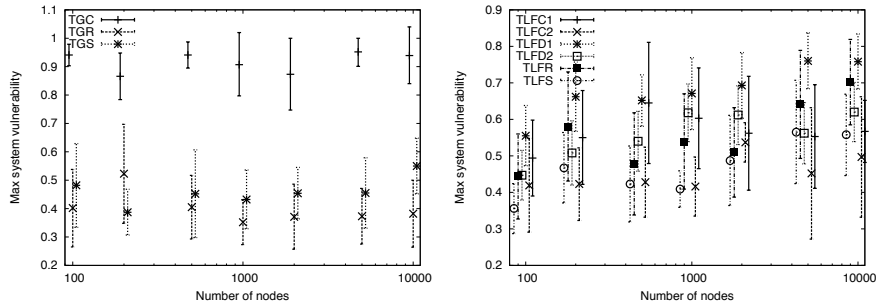


Figure 7: Maximum system vulnerability \mathbf{S} for global (left) and local topologies (right) on a tightly clustered node distribution \mathcal{N}_T .

Figure 7 shows the system vulnerability to node removal for various topologies. Of the global topologies the \mathcal{TG}_C is by far the most vulnerable. The random topology \mathcal{TG}_R is by comparison very resilient although the small world topology \mathcal{TG}_S is almost as resilient. For the local topologies the picture is less clear. The delay based topology \mathcal{TLF}_{D1} is the least resilient. This is because this topology will tend to arrange itself along low delay trees. The introduction of a requirement for diversity in connections in \mathcal{TLF}_{D2} reduces this vulnerability somewhat. It should be noted that most of the confidence intervals in this graph are very large. The system vulnerability can change greatly with each run using the same parameters.

Figure 8 shows similar but not identical results using the metric \mathbf{V} which attempts to measure how vulnerable each node is to the removal of single nodes from the network. In this case, \mathcal{TG}_C performs terribly as does \mathcal{TLF}_{C1} , \mathcal{TLF}_{D1} and to a lesser extent \mathcal{TLF}_{C2} . The random topologies (\mathcal{TG}_R and \mathcal{TLF}_R) obviously do well as these are

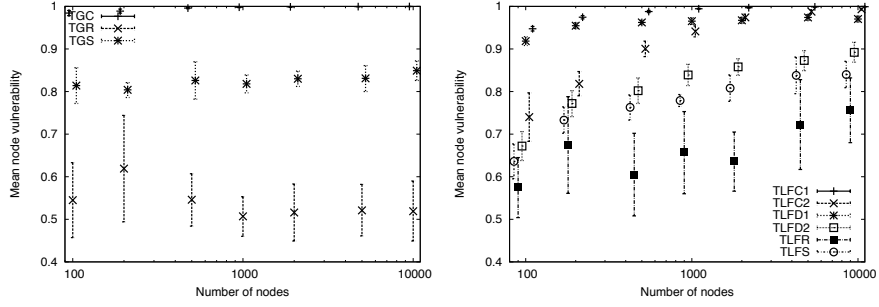


Figure 8: Mean node vulnerability V for global (left) and local topologies (right) on a tightly clustered node distribution \mathcal{N}_T .

unlikely to have single points of failure. The small world topologies \mathcal{TLF}_S and \mathcal{TG}_S also perform relatively well, probably due to their random elements.

4.4 Fairness metric

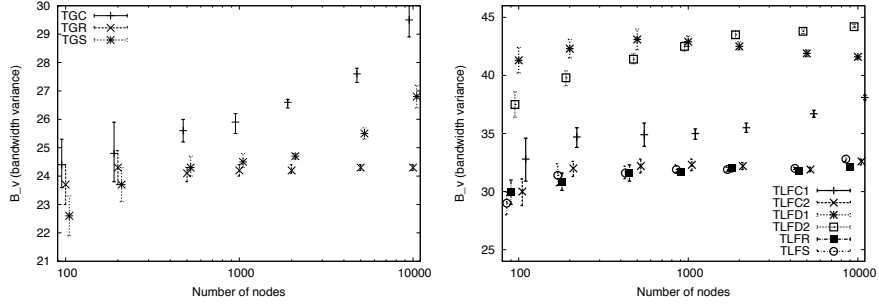


Figure 9: Bandwidth variance B_v for global (left) and local topologies (right) on a tightly clustered node distribution \mathcal{N}_T .

The results on bandwidth variance are shown in figure 9. A low variance would indicate nodes sharing upload capacities “fairly” (although this does not account for node capacities – no clear picture emerged when metric involving proportional bandwidth was used). The topology \mathcal{TG}_C has a higher bandwidth variance than the other global topologies. However, the worst performing topologies are clearly \mathcal{TLF}_{D1} and \mathcal{TLF}_{D2} . This is because those topologies are very likely to efficiently exploit peers with low delay connections to the peercaster and such peers are likely to have their entire bandwidth exhausted.

4.5 Tradeoffs in metrics

Figure 10 shows how vulnerability interacts with delay. The plot on the left is of V versus D_{\max} and on the right is S versus D_{\max} . Each point on the plot is for the mean of ten runs for a given topology and a given node distribution. As can be seen, the

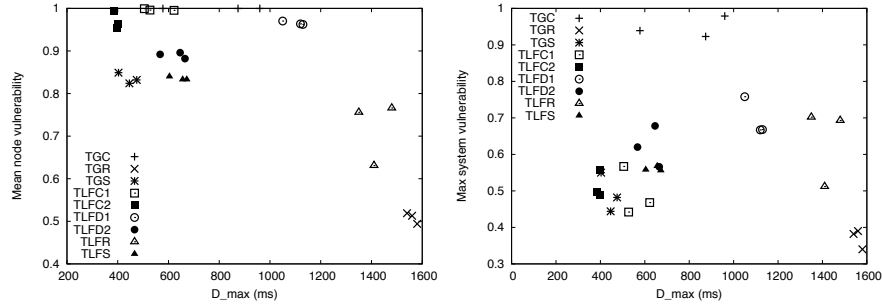


Figure 10: Mean node vulnerability versus V versus D_{\max} (left) and max system vulnerability S versus D_{\max} (right) for all topologies (using all three node distributions).

results are not particularly sensitive to the node distribution used apart from for the topology TG_C .

The best performing topologies are those to the bottom left of the graphs. For the system vulnerability measure S then the best topologies seem to be TLF_{C2} and TG_S and to a lesser extent TLF_S and TLF_{C1} . For the node vulnerability measure V then the best topologies seem to be TG_S or, if resilience is very much more important than delay, TG_R . Of the local topologies, TLF_S , TLF_{D2} and TLF_{C2} perform best with TLF_R being preferred only if delay is much less important than network resilience.

4.6 The effects of the node distribution

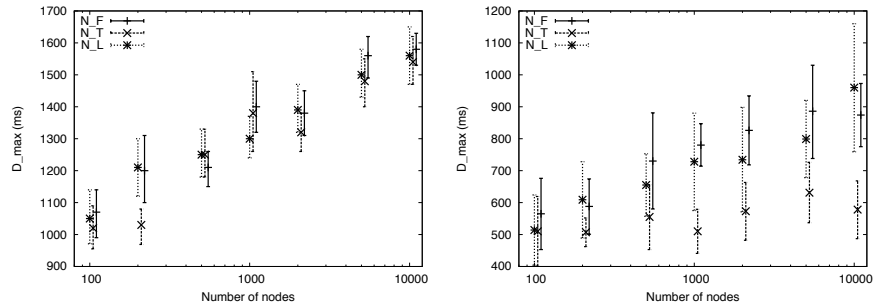


Figure 11: Maximum delay D_{\max} versus number of iterations for topology TG_R (left) and TG_C (right) showing the effects of node distributions \mathcal{N}_F , \mathcal{N}_T and \mathcal{N}_L .

Figure 11 shows the effect of node distribution on delays. The plots are of the number of nodes versus the maximum delay D_{\max} for a given topology. Each line represents one of the three methods for creating a node distribution \mathcal{N}_F , \mathcal{N}_T and \mathcal{N}_L (a flat distribution, a tightly clustered distribution and a loosely clustered distribution). The differences between the distributions (as seen in figure 1) is quite marked. As can be seen, for the random distribution TG_R the impacts of the node distributions on the metrics chosen were minimal (in almost all cases the 95% confidence intervals overlap). This seems to be a fairly typical result. This can be seen by the fact that in figure 10 the

three points for a given topology are almost always clustered together indicating that the topology is of much greater importance than the choice of node distribution used. The exception is for the \mathcal{TG}_C topology. This topology is its closest connected neighbour from all the nodes which will ever enter the system. This seems to be the only case where node clustering makes a marked difference to the results used. However, it should be noticed that even in this case the 95% confidence intervals overlap for most experiments.

5 Conclusions and further work

It is clear that much work remains to be done on this topic, however, some clear conclusions can be drawn. Naive policies like “connect to my closest peer” are not as effective as might be thought in reducing delay to the peercaster. This can produce systems which have a high delay from the peercaster to the peer and also with a high vulnerability to churn. This problem can be mitigated by the so-called “small-world” topologies used here where most connections are local but some are distant.

For the delay measures investigated, some of the topology methods used seem to scale very well indeed with system size. The global small world topology seemed almost delay invariant as the number of nodes in the system increased. Of the local topologies, the topology which tried “hardest” to minimise delay (with nodes aggressively using all the bandwidth of nodes to which they had the lowest delay connection to the peercaster) fared surprisingly badly both in terms of delay and in terms of vulnerability.

With global system knowledge the small world topology (three close connections and one random) performed extremely well in terms of both delay and vulnerability. For the local topology policies the local small world policy or the local closest with diversity policy seemed to offer the best trade off.

An interesting outcome of this research is that, for the parameters used here, the system seemed extremely insensitive to the node distribution used. The node distribution policies were chosen so that the nodes were laid out in a delay space of approximately the same size. However, only for the global closest topology policy were significant differences found in metrics due to a change in the node distribution. This is important since, if this conclusion is more widely applicable, it could free modellers from the (possibly extremely time consuming) task of attempting to validate a peer-to-peer model against a realistic distribution of global delay.

Much remains to be done to complete this work and there are many further avenues which could be investigated. The metrics used could be improved (although at the expense of computational complexity). The global topology methods used could sometimes generate unrealistic topologies (in the sense that a max-flow/min-cut would find that the network had a max flow less than the stream total bandwidth). This could be improved by adding constraints on the global topology creation (a node could only have n uploads if it already had n downloads and had capacity for at least n uploads). There are many other simulation parameters which could be investigated. The choice of four streams here and the distribution of upload capacities was somewhat arbitrary. However, it is difficult to run simulations with too many “degrees of freedom”. A repeated experiment with only one node distribution topology but differences in the distributions of upload bandwidths might generate some interesting results. Indeed a large problem with this research is that the state space to explore is extremely large even in this simple simulation.

An obvious next stage is to allow the local topologies a “reevaluation” stage. That is

to say, that after peers had chosen their upload connections, a second selection process would allow peers to swap to a different uploader. This adds complications since it might be desirable to allow a peer to “displace” another downloader, however, this might create problems if the displacement caused other problems with the topology (for example by cutting off sections of the network from the peercaster).

References

- [1] R. Bindal, Pei Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 66–66, 2006.
- [2] Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, distributed network coordinates. *SIGCOMM Comput. Commun. Rev.*, 34(1):113–118, 2004.
- [3] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.*, 34(4):15–26, 2004.
- [4] Hrishikesh Deshpande, Mayank Bawa, and Hector Garcia-Molina. Streaming live media over a peer-to-peer network. Technical Report Stanford Database Group 2001-20, CS Department, Stanford University, August 2001.
- [5] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.
- [6] Jonathan Ledlie, Paul Gardner, and Margo I. Seltzer. Network coordinates in the wild. In *NSDI*. USENIX, 2007.
- [7] Eng Keong Lua and Xiaoming Zhou. Bos: Massive scale network-aware geometric overlay multicast streaming network. *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 253–258, Nov. 2007.
- [8] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking, 2002.
- [9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection, 2002.
- [10] D. Ren, Y.-T. Li, and S.-H. Chan. On reducing mesh delay for peer-to-peer live streaming. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1058–1066, April 2008.
- [11] Yuval Shavitt and Tomer Tankel. The curvature of the Internet and its usage for overlay construction and distance estimation. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages – 384. Proc. of IEEE Infocom, April 2004., 2004.

Terminating Passage-Time Calculations on Uniformised Markov Chains

Allan Clark* Stephen Gilmore†

Abstract

Uniformisation[1, 2] is a key technique which allows modellers to extract passage-time quantiles/densities which in turn permits the plotting of probability density and cumulative distribution functions. Uniformisation converts a CTMC (Continuous-Time Markov Chain) into a DTMC (Discrete-Time Markov Chain) with equivalent semantics. This can be used to calculate the probability of completing a passage within a given time t by calculating the probability of completing the passage within a number of iterations, n , of the DTMC and then calculating the probability that the n th iteration is performed within time t . However to calculate the passage-time quantiles we desire we must theoretically perform this calculation for values of n from zero to infinity and sum the probabilities. This can be approximated by calculating for values of n from zero to some finite value if we know that larger values of n will yield negligible probabilities and hence add nothing significant to the summation. This paper discusses two important conditions which ensure that the approximation is appropriate while also reducing the amount of negligible values calculated.

1 Introduction

Passage-time quantiles are often desirable measurements to be made from a performance model. In the case of a passage-time measurement we measure from a set of source states to a set of target states. A passage-time quantile is simply a point taken along the cumulative distribution function (cdf) of the passage in question. The cumulative distribution function maps time (usually along the x-axis) against the probability of completing the passage at or before that time. This allows the modeller to answer such questions as: “What is the probability that a request is responded to within 4 seconds?” It is also possible to plot the probability density function (pdf) where the cdf is the integral of the pdf. The pdf then maps time against the probability density of completing the passage at exactly that time.

This paper describes the calculation of passage-time quantiles/densities from a CTMC based model. The technique used is known as *uniformisation* – though it is sometimes called *randomisation*. This paper is chiefly concerned with the steps which follow the actual uniformisation of a CTMC. This is the process

*LFCS, University of Edinburgh, a.d.clark@ed.ac.uk

†LFCS, University of Edinburgh, stg@inf.ed.ac.uk

of extracting from the uniformised Markov chain the probabilities which we desire. For this reason our first example concerns a CTMC which is already in a uniformised state due to all of the rates involved being equal.

The paper is organised as follows; in Section 2 we give an overview of the whole process of uniformisation. Section 3 then introduces an example uniform CTMC which is used to analyse the process of extracting our quantiles from the uniformised Markov chain. Section 4 compares with existing approaches, Section 5 details some finer implementation points and finally in Section 6 we conclude.

The major contribution of this paper is the identification of two properties which allow the accurate halting of the calculation of quantiles. However this paper is sufficiently detailed to serve as an introduction to the key technique of uniformisation in general.

2 Uniformisation

This section details the steps used to derive the cdf (and/or pdf) from a model represented as a CTMC. We first detail the prerequisites:

- The CTMC may be represented by the generator matrix Q . The generator matrix is an $n \times n$ matrix where n is the number of states in the Markov chain. Each row corresponds to one state and the value in one cell of a row corresponds to the rate at which the Markov chain may transition from the state given by the row number to the state given by the column number. The diagonal values are given by subtracting from zero the sum of the other values in the row. Hence if we write $r(i, j)$ to mean the rate at which the Markov chain may transition from state i to j then the generator matrix Q is written as:

$$Q_{i,j} = \begin{cases} r(i,j) & : i \neq j \\ 0 - (\sum_{k=0}^{n-1} r(i,k)) & : otherwise \end{cases}$$

This requires that for any state i , the rate $r(i, i)$ is zero – this condition is usually stated by insisting that the model contains no self-loops.

- The generator matrix may be solved to obtain the steady-state probability distribution π where π_i is the long-term probability of being in state i . This requires that the model be deadlock-free.
- The set of source states \mathcal{S} and the set of target states \mathcal{T} . The probability at time t that we compute is the probability of moving from any of the states in \mathcal{S} to any of the states in \mathcal{T} within time t .

The steps in the computation of the pdf and the cdf of a particular passage within a CTMC are summarised as follows:

- Uniformise the generator matrix to obtain the new matrix P by:

$$P = Q/q + I$$

where Q is the generator matrix, I is the identity matrix and q is a rate value which is chosen to be greater than the magnitude of all of the rates within the generator matrix including the values along the diagonal.

Therefore we have $q > \max_{ij} |Q_{ij}|$ which can be reduced to $q > \max_i |Q_{ii}|$ since the magnitude of the diagonal values in each row are the sums of the other values in the row which cannot be negative. Since q is of greater magnitude than any of the (negative) diagonal values dividing by q returns a negative number $x : -1 < x < 0$. This means that adding the identity matrix ensures that all rate values are positive.

- Add to this uniformised matrix P an absorbing state. This state has no out-going edges to any state other than itself which it loops to with probability 1.
- Modify all target states (states in \mathcal{T}) to transition with probability one to the absorbing state. Call this new matrix P' . The reason for our absorbing state is to ensure that we compute the probability of the first passage and not subsequent completions of the passage. That is; if we are in state $i \in \mathcal{T}$ at time t then we know we are completing the passage at time t and it is not the case that we completed the passage at some earlier time and remained in or returned to state i .
- Compute the probability distribution after n hops of the uniformised Markov chain; given by $\pi^{(n)}$ where $\pi^{(n+1)} = \pi^{(n)} P'$. And $\pi^{(0)}$ is computed using the steady-state probabilities of the source states by:

$$\pi_k^{(0)} = \begin{cases} 0 & : k \notin \mathcal{S} \\ \pi_k / \pi_{\mathcal{S}} & : k \in \mathcal{S} \end{cases}$$

where π_k is the steady-state probability of being in state k and $\pi_{\mathcal{S}}$ is the steady-state probability of being in any one of the source states, that is $\sum_{k \in \mathcal{S}} \pi_k$. Where there is exactly one source state then the steady-state probability distribution need not be calculated and $\pi^{(0)}$ is given by:

$$\pi_k^{(0)} = \begin{cases} 0 & : k \notin \mathcal{S} \\ 1 & : k \in \mathcal{S} \end{cases}$$

since for the one source state j , $\pi_j = \pi_{\mathcal{S}}$.

- For each time t compute: $\sum_{n=0}^{\infty} Er_t^{(n)} \pi_{\mathcal{T}}^{(n)}$ where $Er_t^{(n)}$ is the probability that the n th hop will be performed at or before time t and $\pi_{\mathcal{T}}^{(n)}$ is the probability of being in any of the target states after exactly n hops of the uniformised matrix, P' .

In the final step above we have computed the cdf of the passage by multiplying the probability of being in a target state after exactly n hops by the probability of performing n within the time t . This probability is given by: $\left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right)$. For the pdf we substitute this for the probability of performing n hops at exactly time t . This is given by: $\frac{q^n t^{n-1} e^{-qt}}{(n-1)!}$.

For completeness we provide the full formulae for computing the cdf and pdf of the passage respectively given by:

$$F_{i\vec{j}}(t) = \sum_{n=1}^{\infty} \left(\left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right) \sum_{k \in \vec{j}} \pi_k^{(n)} \right)$$

and

$$f_{i\vec{j}}(t) = \sum_{n=1}^{\infty} \left(\frac{q^n t^{n-1} e^{-qt}}{(n-1)!} \sum_{k \in \vec{j}} \pi_k^{(n)} \right)$$

Transient Measures A transient measure seeks to obtain information about a model from the initial state of the model. The intention is to answer such questions as “What is the expected time before the server first breaks?” Because we are asking about the model in the short term and not the long term the steady-state distribution need not be calculated. The measurement reduces to a passage-time measurement in which there is only one source-state (namely the initial system configuration). A transient measure is often of use if the model is not free from deadlock – asking the probability that the system is deadlocked after a given time is a common transient measure. Depending on the particular query the transient measure may or may not require the addition of the absorbing state. In this paper we focus on passage-time measurements.

3 Snakes And Ladders

In this section we detail an example Markovian analysis of the simple board game “Snakes and Ladders”. Before we proceed with the analysis a brief revision of the rules. Players start by placing a counter on the start square and take it in turns to roll a dice. When a player roles the dice they move their counter forward the number of squares equal to the number they have rolled on the dice. If a player lands directly on a square on which rests the bottom of a ladder they can immediately move their counter to the square at the top of the ladder. Similarly if a player lands directly on a square on which rests the head of a snake then that player must move their counter down to the square at the tail of the snake.

Analysing this game using a DTMC to assess the percentage chance of winning any game within a number of turns N has already been done. If we wish to analyse how long in wall-clock time it will take to complete a game then we must combine the information about how likely it is to win the game after N turns together with the probability of performing N turns within a given time.

So our situation is exactly as in the case that we had started with a CTMC and used uniformisation to obtain a uniform CTMC except that our CTMC was already neatly uniformised to begin with.

We will use a simplified version of the game with only sixteen squares and a dice with only three sides, a player can only roll a 1, a 2 or a 3. We further simplify our task by analysing how long we can expect one player playing by themselves to complete the game.

The game board looks like the one drawn in Figure 1.

The DTMC representing this can be shown in Table 1, note that there are only thirteen game states as opposed to sixteen, this is because the state representing square 5 is equivalent to the state representing square 12 since when a player lands on square 5 their token is automatically moved to square 12. In particular notice that the states representing squares 2, 3 and 4 each have an out-going edge straight to square 12 rather than square 5. Should a player on square 2 roll a 3 their token will end up on square 12, no token can therefore rest on square 5 so we omit it from the state space. Similarly for the two squares, 13 and 14, on which there is a snake. There are fourteen states in total because the absorbing state which we will require is already shown. In the table transitions representing a move to a ladder or snake square and the resulting jump have their rates written in green and red respectively.

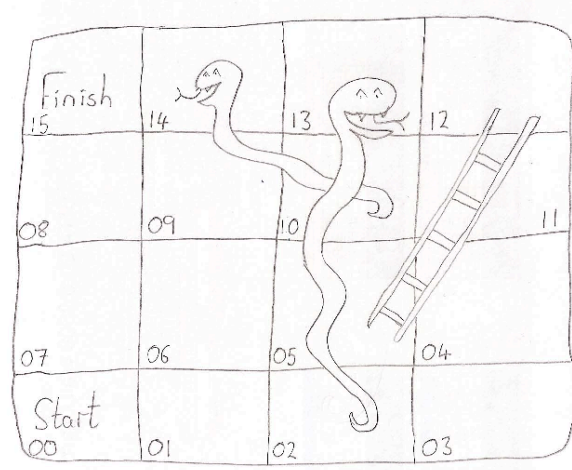


Figure 1: The simplified Snakes and Ladders board game.

	0	1	2	3	4	6	7	8	9	10	11	12	15	Abs
0		$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$										
1			$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$									
2				$\frac{1}{3}$	$\frac{1}{3}$									
3					$\frac{1}{3}$	$\frac{1}{3}$								
4						$\frac{1}{3}$	$\frac{1}{3}$							
6							$\frac{1}{3}$	$\frac{1}{3}$						
7								$\frac{1}{3}$	$\frac{1}{3}$					
8									$\frac{1}{3}$	$\frac{1}{3}$				
9										$\frac{1}{3}$	$\frac{1}{3}$			
10			$\frac{1}{3}$								$\frac{1}{3}$			
11			$\frac{1}{3}$									$\frac{1}{3}$		
12			$\frac{1}{3}$										$\frac{1}{3}$	
15														1
Abs														1

Table 1: The DTMC of a simple snakes and ladders game

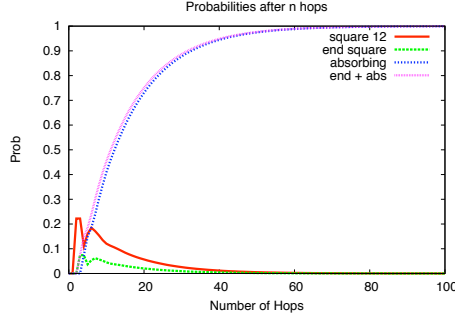


Figure 2: The probability of completing the game after N hops

We can measure from the set of source states: $\{0\}$ to the set of target states: $\{15\}$, note that this means we do not require a steady-state distribution since there is exactly one source state. The graph in Figure 2 shows the probability of being in particular states after N hops. The probability flows from the other states through the target state and into the absorbing state which is why the probability of being in the absorbing state increases as N increases and will equal one in the limit. The line for the target state depicts the probability density function (against number of hops rather than time). A fourth line is drawn which adds the probability of being in either the end state or the absorbing state. This depicts the cumulative distribution function of the game as it adds the probability completing the passage at exactly N hops (the probability of being in the target state) to the probability of completing the game before N hops (the probability of being in the absorbing state).

Now that we know the probability of completing the game in exactly N hops, we may use this information to calculate the probability of completing the game at or by a given time. Since in our example each hop represents one turn or move of the game then we need only know the rate or average duration of a turn in the game. If we assume that each turn lasts about six seconds then the rate at which they occur is ten-per-minute. The graph (Fig. 3, left) shows the probability of performing exactly N hops at or by time t for various values of t . The graph (Fig. 3, right) shows the probability of performing N hops at or by time t seconds for various values of N .

3.1 Producing our final cdf

We can now combine the information in the graphs (Fig 2) and (Fig. 3, right) to produce the cumulative distribution function for the time it will take one player to complete the simplified snakes and ladders game. We know the probability of completing the passage in exactly N hops, $\pi_{\mathcal{T}}^{(n)}$, for all values of N . In addition we know the probability of performing N hops in a given amount of time t , $Er_t^{(n)}$. If we multiply these two values for a given N and a given t this gives us the probability of completing the game in the given t using exactly N hops – this value we designate as $P_n(t)$. Therefore to compute the general probability of completing the game within the given amount of time t we need to sum up

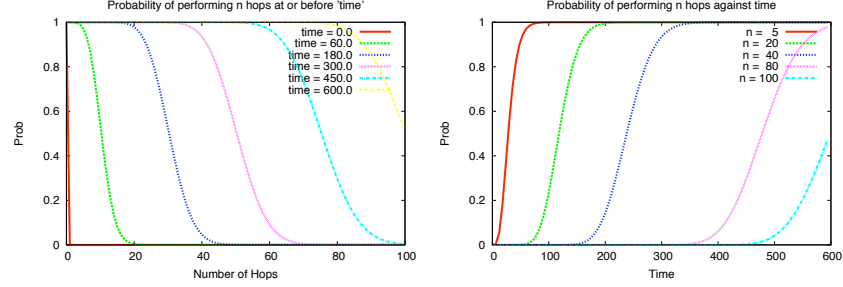


Figure 3: Graphs showing the probability of performing a number of hops by a given time.

Probability of:	value	name
completing the passage in exactly N hops	$\sum_{k \in J} \pi_k^{(n)}$	$\pi_T^{(n)}$
performing N hops within time t	$\left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!}\right)$	$Er_t^{(n)}$
completing the passage in N hops by time t	$\pi_T^{(n)} Er_t^{(n)}$	$P_n(t)$
completing the passage by time t	$\sum_{N=0}^{N=\infty} P_n(t)$	cdf

Table 2: Relationships between the probability values

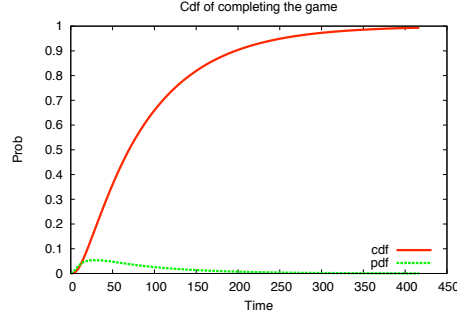
$P_n(t)$ for all values of N from zero to infinity. The relationships between these probabilities are shown in Table 2.

Clearly, summing all of these probability values from zero to infinity is impossible for a computer to do. However there will be some value X for which all values $N_X > X$ the probability of completing the passage within the given time in exactly N_X hops is negligible. Hence at this point we may stop computing probability values. The main contribution of this paper is determining the two conditions which suffice to find the value X .

Previously one method was to compute the probabilities for successive values of N and whenever the probability ($P_n(t)$) was sufficiently low we assume that subsequent values will also be sufficiently low. This method has problems when the passage we must complete has separate paths which vary greatly in their length of hops. In this instance it is possible for the probability to drop below the threshold value but to later climb above it. In this case the given method would stop calculating the probability values before they have a chance to rise above the threshold once more.

Our method is to monitor the probability of being in the absorbing state after N hops – we designate this value $Abs^{(n)}$. When this value climbs to within a suitable threshold of 1 then there is no probability left to flow through the target states. Hence the probability of being in a target state for all values of N greater than the current value must be below the threshold value, since this probability is multiplied by the probability of performing N hops within time t we know that all subsequent probabilities will be below the threshold.

This method performs well, however, for small values of t we find that we compute more hop-values than are required. This is because for small values of

Figure 4: The probability of completing the game after N hops

t it is unlikely that we are able to perform a large number of hops. However if the passage is long then it may be that the probability of being in the absorbing state does not climb to within the threshold of one until N is large – whereby ‘large’ we mean “larger than the number of hops we could hope to perform within the time t ”. Therefore we also monitor the value of $P_n(t)$ – the probability of performing N hops within time t – whenever this value falls below the threshold, we know that any subsequent values of N will yield negligible probability at time t (since $P_n(t)$ is involved in the product to find the probability) and hence we have determined a suitable value of X .

Our algorithm may be summed up by a recursive function as:

$$cdf(n, t) = \begin{cases} 0 & : Abs^{(n)} > (1 - threshold) \\ 0 & : Er_t^{(n)} < threshold \\ (\pi_{\mathcal{T}}^{(n)} * Er_t^{(n)}) + (cdf(n+1, t)) & : otherwise \end{cases}$$

and similarly for the pdf function:

$$pdf(n, t) = \begin{cases} 0 & : Abs^{(n)} > (1 - threshold) \\ 0 & : Er_t^{(n)} < threshold \\ (\pi_{\mathcal{T}}^{(n)} * Erp_t^{(n)}) + (pdf(n+1, t)) & : otherwise \end{cases}$$

Where $Erp_t^{(n)}$ is the probability of performing the N th hop at exactly time t and is given by: $\frac{q^n t^{n-1} e^{-qt}}{(n-1)!}$. Since there is a lot of shared computation our implementation computes both the cdf and the pdf of the passage together.

Finally then we may draw our graphs of the cumulative distribution and probability density functions of our snakes and ladders game. These are depicted in Figure 4.

4 Comparison with existing techniques

The naïve approach which we briefly illustrated in section 3.1 is to compute values for successive values of N until such values drop below a threshold. This method may be summed up by:

$$cdf(n, t) = \begin{cases} 0 & : (\pi_{\mathcal{T}}^{(n)} * Er_t^{(n)}) < threshold \\ (\pi_{\mathcal{T}}^{(n)} * Er_t^{(n)}) + (cdf(n+1, t)) & : otherwise \end{cases}$$

As we mentioned above this algorithm suffers from a problem if the input passage has multiple paths to completion of varying lengths. In this case the value at some N may drop below the threshold but may later rise above the threshold again. The simple solution would stop after the first time it drops below the threshold.

As an improvement on this technique the Markovian response-time analyser Hydra[3, 4, 5, 6] monitors the value of the erlang distribution with a q rate parameter and N hop parameter. The Hydra solution can therefore be summarised by:

$$cdf(n, t) = \begin{cases} 0 & : Er_t^{(n)} < threshold \\ (\pi_T^{(n)} * Er_t^{(n)}) + (cdf(n+1, t)) & : otherwise \end{cases}$$

Therefore this solution will compute the same values as our solution in all cases because our solution contains the same condition. However our solution is a further refinement which allows us to avoid needless computation for some values of N . In particular where the t-range — that is the times for which we should compute the passage-time quantiles — specified is too large. Suppose the user has specified a t-range of 1 – 1000 but the passage has a probability very close to one of completing by time 500. Because there is a large probability of completing the passage by time 500 this means that there is a large probability of completing the passage within a number of hops X and that X hops are very likely to be performed within 500 time units. This means that for time values over 500 there will be a possibility to perform more than X hops and the Hydra solution will continue to compute probabilities for these hop values. However our solution would recognise that such values cannot add anything to the cdf because you are very like to have completed the passage before X hops. In the case of the cdf this could be mitigated by incorporating the naïve solution but this is not as effective for computing the pdf.

Our solution has a further, related, advantage; the user need not specify the upper-bound on the t-range at all. The user need only give the start of the t-range and the steps in which we wish to increase the value of t . This is because using our technique we can calculate the value X at which performing more than X number of hops will not significantly add to the probability of completing the passage (because there is a probability within the threshold of one of being in the absorbing state by X hops). We can then use this to work out the upper-bound on the t-range by calculating the value of t such that performing X hops within time t is significantly likely. In order that the user need not specify a t-range at all we default to a starting time of zero and a time-step of the calculated stop-time divided by one hundred. The user may then override any of; the start-time, the stop-time, the time-increments and the number to divide the t-range by in order to obtain the time-increments.

5 Implementation

The techniques described in this paper have been fully implemented in the International PEPA Compiler (ipc) based on the ipclib[7]. This is a compiler for the Performance Evaluation Process Algebra (PEPA)[8], is open source software and may be downloaded from: <http://www.dcs.ed.ac.uk/pepa/tools/ipc/>

5.1 Technical Points

This paper has shown how to compute passage-time quantiles from continuous time Markov chains. However we have left the actual numerical computation as a given, though this is non-trivial. For the cumulative distribution function we must compute:

$$F_{ij}^{\rightarrow}(t) = \sum_{n=1}^{\infty} \left(\left(1 - e^{-qt} \sum_{k=0}^{n-1} \frac{(qt)^k}{k!} \right) \sum_{k \in \mathcal{T}} \pi_k^{(n)} \right)$$

Notice in particular that we must compute $k!$ for what may be large values of k . In addition we must compute qt^k , also for potentially large values of k . The large values here are in the order of the number of hops, if we have large differences between the rates this value may be quite high — a value in the order of thousands is not uncommon (in [9] this number is said to be of the order of qt). Hence we can expect to encounter a problem with overflow. Even if some arbitrary precision library is used (at a performance cost) computing the cdf in this way is inefficient. Our first observation is that:

$\frac{(qt)^k}{k!}$ is equal to: $\prod_{i=1}^{i=k} \frac{qt}{i}$
which allows us to avoid the computation of the large power and factorial values.

Now for each value of N we must compute $\sum_{k=0}^N \frac{(qt)^k}{k!}$, we need not compute each term separately we can instead compute the infinite list of values by the recursive function:

$$\begin{aligned} \text{sumvalues}(n, \text{current}) &= \text{current} : \text{rest} \\ \text{where rest} &= \text{sumvalues} \left(n + 1, \text{current} + \sum_{k=0}^N \frac{(qt)^k}{k!} \right) \end{aligned}$$

Because our implementation is in the lazy programming language Haskell we need not worry about the computation of an infinite list since we will only ever examine a finite number of elements from it. For a strict language this laziness can be easily simulated. We now observe that even this computation does a large amount of re-computation. Namely the successive values of $\sum_{k=0}^N \frac{(qt)^k}{k!}$ recompute all previous values. However we can use the same trick:

$$\begin{aligned} \text{prodvalues}(k, \text{current}) &= \text{current} : \text{rest} \\ \text{where rest} &= \text{prodvalues} \left(k + 1, \text{current} \times \frac{qt}{k} \right) \end{aligned}$$

This means that we can now update our *sumvalues* function to take advantage of this. It now becomes a list transformation function which takes in the list of product values computed by the above *prodvalues* function.

$$\begin{aligned} \text{sumvalues}(\text{current}, (n, p) : \text{rest}) &= (n, \text{current}) : \text{restsum} \\ \text{where restsum} &= \text{sumvalues}(\text{current} + p, \text{rest}) \end{aligned}$$

We can also factor out the code to calculate the probability of being in the absorbing state and/or a target state after exactly N hops. Since otherwise we will recompute these values for each time value we desire. Once we have factored out all the common computation we have a set of infinite lists which map N from $N = 0$ to $N = \infty$ to values used in the computation of the cdf and pdf. We need only operate on these lists for the values of t .

5.2 Computing hops

We must compute the probability of completing the passage in a given number of hops. Recall that the probability of completing the passage in exactly N hops is given by: $\sum_{k \in \mathcal{T}} \pi_k^N$

Further recall that each hop is computed via the previous hop as:

$$\pi^{(n+1)} = \pi^{(n)} P'$$

Therefore we are required to perform successive matrix multiplications. If we have a large matrix P' these matrix multiplications may be expensive. However we note that P' is a modified version of the matrix P which is a uniformised version of the original generator matrix Q . We have modified P by adding an absorbing state and each state in the target set \mathcal{T} has been mutated to target only the absorbing state. This means that, even in the case that all states in the matrix Q were reachable there may be a set of states which are now unreachable, call this set \mathcal{U} . In fact this set is likely to be non-empty. It represents all the states in the original which are not on any path from \mathcal{S} to \mathcal{T} but are on some path from \mathcal{T} to \mathcal{S} .

Consider the model of a system containing 20 clients and 2 servers. Each server may accept *requests* and subsequently make a *response*. The servers therefore have two states: *Available* and *Busy*. Each client synchronises with one of the two servers over a *request* and then waits for a *response*. Between subsequent communications with a server each client must do two pieces of work. The client therefore has three states which it cycles through; *Working*, *Waiting* and *Using*. The *Using* state corresponds to the using of the data returned by the server and the *Working* state corresponds to the generation of a new request to the server.

If we wish to measure the response-time of this model we must measure the response-time as observed by a single client, suppose we choose the first client named *Client*₀. The set of source states is the set of states in which the first client is in the *Waiting* state and is a target of a transition in which the source state of the transition has the first client in the *Working* state. Conversely the set of target states is the set of states in which the first client is in the *Using* state and is a target of some transition from a state in which the first client is in the state *Waiting*.

For this measurement all states in which the first client is in the *Waiting* state are along some path from \mathcal{S} to \mathcal{T} however all states in which the first client is in the *Working* state are not in \mathcal{T} (since the client must go through the *Using* state) or in \mathcal{S} or on some path between \mathcal{S} and \mathcal{T} . These states are all in the unreachable set \mathcal{U} . In this particular case this corresponds to approximately half of the entire state space of the model. Where there are more client states not within the passage this ratio can increase such that the size of the set \mathcal{U} is much larger than the set of states not in \mathcal{U} .

Because we know that for any n and $i \in \mathcal{U}$:

$$\pi_i^{(n)} = 0$$

we can avoid a lot of calculation by transforming the matrix P' into a smaller matrix removing the unreachable states. Since we perform many matrix multiplications using this matrix to obtain the hops, this is a potentially very large saving.

6 Conclusions

In this paper we have given a detailed account of the calculation of passage-time quantiles and densities from continuous-time Markov chains. Although we have shown how to obtain the uniformised matrix from the original generator matrix of the CTMC, we have focussed on the calculations that occur once the matrix has already been uniformised. To this extent our main contribution has been

the identification of two important properties which allow the otherwise infinite calculation to terminate. The two properties have the desired feature that we are conservative — meaning that we never terminate too early producing an erroneous answer. However the combination of the two provides early detection to avoid some needless calculations. In addition we feel that our paper is a good introduction to the topic of uniformisation in general.

Acknowledgements: The authors are supported by the EU FET-IST Global Computing 2 project SENSORIA (“Software Engineering for Service-Oriented Overlay Computers” (IST-3-016004-IP-09)) and the EPSRC PerformDB project (EP/D054087/1). The ipc/Hydra tool chain has been developed in co-operation with Jeremy Bradley, Will Knottenbelt and Nick Dingle of Imperial College, London.

References

- [1] Grassmann, W.: Transient solutions in Markovian queueing systems. *Computers and Operations Research* **4** (1977) 47–53
- [2] Gross, D., Miller, D.: The randomization technique as a modelling tool and solution procedure for transient Markov processes. *Operations Research* **32** (1984) 343–361
- [3] Dingle, N.J., Knottenbelt, W.J., Harrison, P.G.: HYDRA: HYpergraph-based Distributed Response-time Analyser. In Arabnia, H.R., Man, Y., eds.: PDPTA’03, Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications. Volume 1., Las Vegas, NV (2003) 215–219
- [4] Knottenbelt, W.J.: Generalised Markovian Analysis of Timed Transition Systems. Master’s thesis, Department of Computer Science, University of Cape Town (1996)
- [5] Dingle, N.J.: Parallel Computation of Response Time Densities and Quantiles in Large Markov and Semi-Markov Models. PhD thesis, Department of Computing, Imperial College London. University of London. (2004)
- [6] Bradley, J., Dingle, N., Gilmore, S., Knottenbelt, W.: Extracting passage times from PEPA models with the HYDRA tool: A case study. In Jarvis, S., ed.: Proceedings of the Nineteenth annual UK Performance Engineering Workshop, University of Warwick (2003) 79–90
- [7] Clark, A.: The ipclib PEPA Library. In Harchol-Balter, M., Kwiatkowska, M., Telek, M., eds.: Proceedings of the 4th International Conference on the Quantitative Evaluation of SysTems (QEST), IEEE (2007) 55–56
- [8] Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996)
- [9] Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model checking continuous-time markov chains by transient analysis. In: CAV ’00: Proceedings of the 12th International Conference on Computer Aided Verification, London, UK, Springer-Verlag (2000) 358–372

Which battery model to use?

Marijn R. Jongerden* Boudewijn R. Haverkort†

Abstract

The use of mobile devices like cell phones, navigation systems, or laptop computers, is limited by the lifetime of the included batteries. This lifetime depends naturally on the rate at which energy is consumed, however, it also depends on the usage pattern of the battery. Continuous drawing of a high current results in an excessive drop of residual capacity. However, during intervals with no or very small currents, batteries do recover to a certain extent. The usage pattern of a device can be well modeled with stochastic workload models. However, one still needs a battery model to describe the effects of the power consumption on the state of the battery. Over the years many different types of battery models have been developed for different application areas. The best type of model to use in the setting of performance modelling are analytical models. In this paper we analyse two well-known analytical models, and show that one is actually an approximation of the other; this was not known previously. Furthermore, we tested the suitability of these models for performance evaluation purpose.

1 Introduction

With the proliferation of cheap wireless access technologies, such as wireless LAN, Bluetooth as well as GSM, the number of wireless devices an average citizen is using has been steadily increasing over the last decade. Such devices do not only add to the flexibility with which we can do our work, but also add to our reachability and our security. Next to these personal wireless devices, an ever growing number of wireless devices is used for surveillance purposes, most notably in sensor-type networks. A common issue to be dealt with in the design of all of these devices is power consumption. Since all of these devices use batteries of some sort, mostly rechargeable, achieving low power consumption for wireless devices has become a key design issue. This fact is witnessed by many recent publications on this topic, and even a special issue of *IEEE Computer* devoted to it [1].

Low-power design is a very broad area in itself, with so-called “battery-driven system design” a special branch of it, that becomes, due to the reasons mentioned, more and more important. A key issue to be addressed is to find the right tradeoff between battery usage and required performance: how can we design a (wireless) system such that with a given battery, good performance

*University of Twente, Faculty for EEMCS, Centre for Telematics and Information Technology, jongerdenmr@ewi.utwente.nl

†University of Twente, Faculty for EEMCS, Centre for Telematics and Information Technology, brh@ewi.utwente.nl

(throughput, reachability, and so on) is obtained, for a long-enough period. Stated differently, how should the processes in the wireless device be organised such that the battery lifetime (which determines the system lifetime) will be as high as possible. Indeed, it has been observed recently that due to the specific physical nature of batteries, achieving the longest battery lifetime is not always achieved by “just” trying to minimise the power consumption at any point in time. Instead, also the way in which the power is consumed, that is, the current-extraction patterns and the employed current levels play a role in the battery lifetime.

Using an abstract workload model one can model the operation of a system, describing the various states the wireless device can be in, together with the energy consumption rates in those states. Also, the transition possibilities between these states can be represented in the workload model. Such a description can be interpreted as a Markov-reward model in which accumulated reward stands for the amount of energy consumed. The system or battery lifetime then equals the time until a certain level of consumption (the available charge of the battery) is reached. Determining this time, or better, its distribution, could be done with well-known techniques for performability evaluation. However, such an approach does not well take into account the physical aspect of battery operation. Indeed, studies on batteries reveal that the battery depletion rate in general is non-linear in time, and, moreover, also depends on the amount of energy still in the battery. [2, 3] Furthermore, in periods when a battery is not used, subtle but important battery-restoration effects are in place, that apparently refill the battery.

To capture the influence of the power consumption on the battery, a battery model is needed. Over the years, many different types of battery models have been developed for different application areas [3]. For example, the electrochemical models described in [4, 5, 6] are used in battery design. These models describe the battery in its very detail using a set of six coupled differential equations. Another example are the electrical circuit models used in electrical engineering [7], which focus on the electrical properties of the battery. Although these models describe the battery accurately, they are not suitable to be used in the setting of the performance models because of the detailed description, which would make the combined model unmanageable. What is needed, is an abstract model which focuses on the important battery properties and their effects only. Two analytical models are good candidates: the Kinetic Battery Model (KiBaM) by Manwell and McGowan [8, 9, 10] and the diffusion based model by Rakhmatov and Vrudhula [11]. These two models describe the battery using only two differential equations. Although the equations the two models start from are really different, we show by applying a coordinate transformation on the KiBaM, that the KiBaM is actually a first order approximation of the diffusion model. A further theoretical and practical comparison of the two models is made, which leads to the conclusion which model is best to use in the setting of the performance models.

The rest of the paper is structured as follows. Section 2 gives a short introduction to battery properties that have to be addressed. In Section 3 the two analytical battery models are introduced and Section 4 gives a theoretical and practical comparison of these models, which leads to the conclusion which model is best to use. In Section 5 the limitations of the analytical model are analysed. We end with conclusions and an outlook to future work in Section 6.

2 Batteries

The two most important properties of a battery are its voltage (expressed in volts V) and its capacity (mostly expressed in Ampere-hour, Ah); the product of these two quantities is a measure for the energy stored in the battery. For an ideal battery the voltage stays constant over time until the moment it is completely discharged, then the voltage drops to zero. The capacity in the ideal case is the same for every load for the battery. Reality is different, though: the voltage drops during discharge and the effectively perceived capacity is lower under a higher load. This phenomenon is termed the *rate capacity effect*.

In the ideal case it is easy to calculate the lifetime of a battery. The lifetime (L) in the case of a constant load is the capacity (C) over the load current (I):

$$L = C/I. \quad (1)$$

Due to various nonlinear effects this relation does not hold for real batteries. A simple approximation for the lifetime under constant load can be made with Peukert's law [11]:

$$L = \frac{a}{I^b}, \quad (2)$$

where $a > 0$ and $b > 1$ are constants which depend on the battery. For variable loads ($i(t)$) one can extend this formula by using the average current up until $t = L$:

$$L = \frac{a}{\left(\frac{1}{L} \int_0^L i(t) dt\right)^b} \quad (3)$$

Following (3), all load profiles with the same average would have the same lifetime. Experimentally it can be shown that this is not the case. One of the effects playing an important role here is the *recovery effect* of the battery. This, is the effect that the battery can regain some of its “lost” capacity during idle periods.

3 Battery models

In this section two analytical battery models are discussed. In both models the non-linear effects of the battery are described using two differential equations. In Section 3.1 we present the diffusion model of Rakhmatov and Vrudhula. In Section 3.2 we present the Kinetic Battery Model of Manwell and McGowan. Then, in Section 3.3, we apply a coordinate transformation on the Kinetic Battery Model, which leads the insight that this model is actually an approximation of the diffusion model.

3.1 Rakhmatov and Vrudhula's diffusion model

An analytical battery model based on the diffusion of the ions in the electrolyte has been developed by Rakhmatov and Vrudhula in 2001 [11, 12, 13]. The model describes the evolution of the concentration of the electro-active species in the electrolyte to predict the battery lifetime under a given load. In the

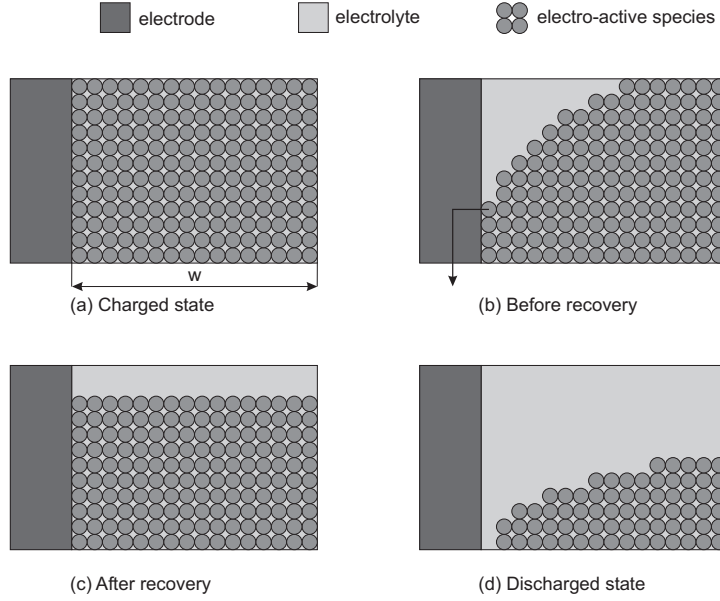


Figure 1: Physical picture of the model by Rakhmatov and Vrudhula

model the processes at both electrodes are assumed identical, thus the battery is assumed symmetric with respect to the electrodes and only one of the electrodes is considered.

Figure 1 shows a simplified view of the battery operation according to the diffusion model. At first, for the full battery, the concentration of the electro-active species is constant over the full width (w) of the electrolyte (Figure 1(a)). When a load is applied to the battery, the electrochemical reaction results in a reduction of the concentration of the species near the electrode. Thus, a gradient is created across the electrolyte (Figure 1(b)). This gradient causes the species to diffuse towards the electrode. Now, when the load is switched off, the concentration of the species at the electrode will increase again (recover) due to the diffusion, and eventually the species will be evenly distributed over the electrolyte again. The concentration, however, will be lower than for the full battery (Figure 1(c)). Finally, when the concentration at the electrode drops below a certain value (C_{cutoff}), the chemical reaction can no longer be maintained and the battery is considered to be empty (Figure 1(d)).

The concentration of the electro-active species at time t and distance $x \in [0, w]$ is denoted by $C(x, t)$. For the full battery the concentration is constant over the length of the electrolyte: $C(x, 0) = C^*$, $x \in [0, w]$. The battery is considered empty when $C(0, t)$ drops below the cutoff level C_{cutoff} . The evolution of the concentration is described by Fick's laws [11]:

$$\begin{cases} -J(x, t) &= D \frac{\partial C(x, t)}{\partial x}, \\ \frac{\partial C(x, t)}{\partial t} &= D \frac{\partial^2 C(x, t)}{\partial x^2}, \end{cases} \quad (4)$$

where $J(x, t)$ is the flux of the electro-active species at time t and distance x from the electrode, and D is the diffusion constant. The flux at the electrode surface ($x = 0$) is proportional to the current ($i(t)$). The flux on the other side of the diffusion region ($x = w$) equals zero. This leads to the following boundary conditions:

$$\begin{cases} D \frac{\partial C(x, t)}{\partial x} \Big|_{x=0} = \frac{i(t)}{\nu F A}, \\ D \frac{\partial C(x, t)}{\partial x} \Big|_{x=w} = 0, \end{cases} \quad (5)$$

where A is the area of the electrode surface, F is Faraday's constant, and ν is the number of electrons involved in the electrochemical reaction at the electrode surface.

It is possible to obtain an analytical solution for this set of partial differential equations (4) together with the initial condition and the boundary conditions 5 using Laplace transforms. From that solution one can obtain an expression for the apparent charge lost from the battery ($\sigma(t)$) [14]:

$$\sigma(t) = \underbrace{\int_0^t i(\tau) d\tau}_{l(t)} + \underbrace{\int_0^t i(\tau) \left(2 \sum_{m=1}^{\infty} e^{-\beta^2 m^2 (t-\tau)} \right) d\tau}_{u(t)}, \quad (6)$$

where $\beta = \pi\sqrt{D}/w$. The apparent charge lost is separated in two parts, the charge lost to the load ($l(t)$) and the unavailable charge ($u(t)$). The first is the charge used by the device. The second is charge which remains in the battery unused. The battery is empty when the apparent charge lost is equal to the battery's capacity.

For a constant current I , (6) can easily be solved. For $l(t)$ one obtains: $l(t) = It$. For the unavailable charge one can interchange the integral and the summation, which leads to:

$$u(t) = 2I \sum_{m=1}^{\infty} \frac{1 - e^{-\beta^2 m^2 t}}{\beta^2 m^2}. \quad (7)$$

During idle periods the unavailable charge will decrease and will be available again for the load. One can compute the function that describes the evolution of the unavailable charge during an idle period after a load I that lasted for a period of length t_l :

$$u(t_i) = 2I \sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 t_i} (1 - e^{-\beta^2 m^2 t_l})}{\beta^2 m^2}, \quad (8)$$

where t_i is the idle time.

3.2 Kinetic Battery Model

Another analytical model which can be used for computing battery lifetimes is the Kinetic Battery Model (KiBaM) of Manwell and McGowan [8, 9, 10]. The

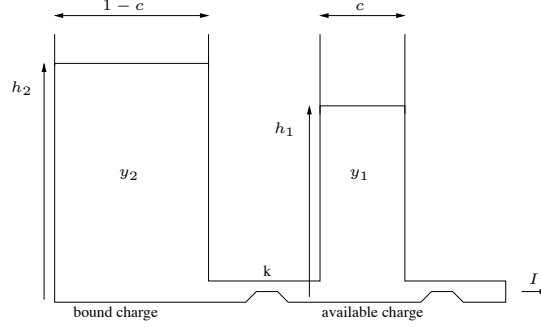


Figure 2: Two-well-model of the Kinetic Battery Model

KiBaM is a very intuitive battery model. In the model the battery charge is distributed over two wells: the available-charge well and the bound-charge well (cf. Figure 2). A fraction c of the total capacity is put in the available charge well, and a fraction $1 - c$ in the bound charge well. The available charge well supplies electrons directly to the load ($i(t)$), whereas the bound-charge well supplies electrons only to the available-charge well. The rate at which charge flows between the wells depends on the height difference between the two wells, and on a parameter k . The heights of the two wells are given by: $h_1 = \frac{y_1}{c}$ and $h_2 = \frac{y_2}{1-c}$. The change of the charge in both wells is given by the following system of differential equations:

$$\begin{cases} \frac{dy_1}{dt} = -i(t) + k(h_2 - h_1), \\ \frac{dy_2}{dt} = -k(h_2 - h_1), \end{cases} \quad (9)$$

with initial conditions $y_1(0) = c \cdot C$ and $y_2(0) = (1 - c) \cdot C$, where C is the total battery capacity. The battery is considered empty when there is no charge left in the available charge well.

When a load is applied to the battery, the available charge reduces, and the height difference between the two wells grows. When the load is removed, charge flows from the bound-charge well to the available-charge well until h_1 and h_2 are equal again. So, during an idle period, more charge becomes available and the battery lasts longer than when the load is applied continuously. In this way the recovery effect is taken into account in the model. Also, the rate capacity effect is covered, since for a higher discharge current the available charge well will be drained faster, less time will be available for the bound charge to flow to the available charge. Therefore, more charge will remain unused, and the effective capacity is lower.

The differential equations (9) can be solved for the case of a constant discharge current ($i(t) = I$) using Laplace transforms, which yields:

$$\begin{cases} y_1 = y_{1,0}e^{-k't} + \frac{(y_{0,k'}c-I)(1-e^{-k't})}{k'} - \frac{Ic(k't-1+e^{-k't})}{k'}, \\ y_2 = y_{2,0}e^{-k't} + y_0(1-c)(1-e^{-k't}) - \frac{I(1-c)(k't-1+e^{-k't})}{k'}, \end{cases} \quad (10)$$

where k' is defined as $k' = k/c(1 - c)$, $y_{1,0}$ and $y_{2,0}$ are the amount of available and bound charge, respectively, at $t = 0$, and $y_0 = y_{1,0} + y_{2,0}$.

3.3 Coordinate transformation

Although the differential equations (9) nicely describe the discharge process of the battery, and an analytical solution can be obtained for constant discharge currents, the equations can be made more simple when a coordinate transform is applied. In this way even more insight can be obtained in the way the model behaves.

From (9) one can see that the height difference between the two wells ($h_2 - h_1$) plays a major role in the model. This is one of the coordinates after the transformation, the other is the total charge in the battery. So, the transformation changes the coordinates from y_1 and y_2 to $\delta_h = h_2 - h_1$ and $\gamma = y_1 + y_2$. This transformation changes the differential equations to:

$$\begin{cases} \frac{d\delta_h}{dt} &= \frac{i(t)}{c} - k'\delta_h, \\ \frac{d\gamma}{dt} &= -i(t), \end{cases} \quad (11)$$

with initial conditions $\delta_h(0) = 0$ and $\gamma(0) = C$. In the new coordinate system the condition for the battery to be empty is: $\gamma(t) = (1 - c)\delta_h(t)$. The differential equations are independent and are straightforwardly solved for constant discharge currents:

$$\begin{cases} \delta_h(t) &= \frac{I}{c} \cdot \frac{1 - e^{-k't}}{k'}, \\ \gamma(t) &= C - It. \end{cases} \quad (12)$$

Like for the diffusion model, one can also compute the evolution of δ_h as a function of the idle time t_i after a load I that lasted for a period t_l :

$$\delta_h(t_i) = \frac{I}{c} \cdot \frac{e^{-k't_i}(1 - e^{-k't_l})}{k'}. \quad (13)$$

The solutions for continuous discharge can be used to obtain a solution for any discharge profile with piecewise constant currents by adapting the initial conditions appropriately. The level of γ and δ_h at the end of a step in the load profile, can be used as initial conditions for the next step.

4 Comparing the analytical models

The interpretation of the diffusion model with its unavailable charge is very similar to the KiBaM with its bound charge. However, it is actually the height difference times $1 - c$ in the KiBaM that plays the same role as the unavailable charge in the diffusion model.

4.1 Continuous discharge

It is possible to write the solution of the transformed KiBaM in the form of the diffusion model with the charge lost split in a load and an unavailable charge part. For constant current discharge this yields:

$$l(t) = C - \gamma(t) = It \quad (14)$$

$$u(t) = (1 - c) \cdot \delta_h(t) = \frac{(1 - c)I}{c} \frac{1 - e^{-k't}}{k'} \quad (15)$$

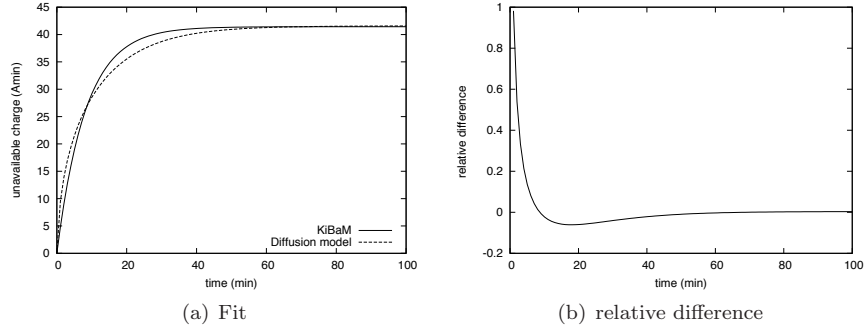


Figure 3: Fit of the KiBaM to the diffusion model. The evolution of the unavailability charge in both the diffusion model and the fitted KiBaM is given in (a). In (b) the relative difference between the two curves is given.

When one compares (15) with (7), one sees that the former has the same form as the first term of the sum of the latter. Setting $c = \frac{1}{3}$ and $k' = \beta^2$ in the KiBaM, results in the first order approximation of the diffusion model. This is of course, a bad approximation of the infinite sum.

One can obtain a much better approximation, when the parameters c and k' are used to fit the KiBaM equation of $u(t)$ to the equation of the diffusion model. Figure 3(a) shows the result of a least squares fitting procedure for the case that $I = 0$. When $\beta = 0.273 \text{ min}^{-\frac{1}{2}}$, the fit results in $c = 0.166$ and $k' = 0.122 \text{ min}^{-1}$. In Figure 3(b) the relative difference between the two curves is shown. This difference is independent of the discharge current. The relative difference is very large, up to 80%, for times smaller than 10 minutes. This implies that the results for battery lifetime computations will differ mainly for high discharge currents.

4.2 Frequency response

Following the method described in [14] an analysis of the frequency response of both the Kinetic Battery Model and Rakhmatov and Vrudhula's diffusion model was done. The results are given in Figure 4. The figure shows that the diffusion model has a higher frequency response for high frequencies. This is due to the high order terms that are included in the diffusion model and not in the KiBaM. However, both models are highly insensitive to high frequency current switching. Currents varying faster than 0.01 Hz can be replaced with an average current without giving significant errors in the battery lifetime computations. The level of the frequency response is mainly determined by the size of the recovery parameter

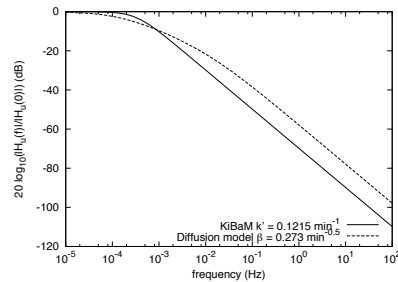


Figure 4: Frequency response for KiBaM and diffusion model

Test	Name	I_{ave} (mA)	DUALFOIL (min)	Diffusion (min)	KiBaM (min)
T1	MPEG	222.7	140.9	139.9	139.9
T2	Dictation	204.5	156.0	156.0	156.0
T3	Talk1	108.3	317.2	331.4	331.4
T4	Talk2	107.5	319.5	334.1	334.1
T5	Talk3	94.9	365.1	384.0	384.0
T6	WAV1	84.3	413.7	437.5	437.5
T7	WAV2	75.5	464.8	493.3	493.3
T8	Idle1	28.0	1278	1400	1401
T9	Idle2	19.5	1852	2029	2029
T10	SleepDC	3.0	12285	13417	13417
T11	IAT	628.0	26	26.6	24.9
T12	IAR	494.7	41.3	41.4	40.5
T13	IST	425.6	54.6	53.9	53.5
T14	ISR	292.3	99.5	96.7	96.7
T15	IAD	265.6	113.1	110.6	110.6
T16	MSD	252.3	120.8	118.6	118.6
T17	DSD	234.1	132.7	131.0	131.0
T18	TSD	137.9	243.6	251.3	251.3
T19	WSD	113.9	300.1	313.0	313.0
T20	ISD	57.6	616.3	659.5	659.5
T21	SSD	32.5	1101	1201	1201
T22	Boot	300.0	96.0	93.2	93.1

Table 1: Lifetimes of continuous current discharge computed with DUALFOIL, the diffusion model (both from [13]) and KiBaM (computed by us). The discharge currents belong to different operational states of the Itsy pocket computer.

(k' or β). An increase of this parameter results in higher frequency response, and thus to a higher sensitivity to fine-grained scheduling.

4.3 Computing lifetimes

Next to the theoretical analysis of the two models, both models were used to compute battery lifetimes for various load profiles.

In [13] Rakhmatov et al. give the battery lifetimes for load profiles of a Compaq Itsy pocket computer, computed both with their diffusion model and the electro-chemical model DUALFOIL [15]. To these results the lifetimes according to the KiBaM model have been added in Table 1 for constant loads and Table 2 for variable-load profiles. Details of the variable-load profiles are given in Table A (in Appendix A).

The lifetimes computed using the KiBaM and diffusion model match very well. The results for continuous discharge only deviate at high discharge currents, as expected from the analysis of the equations, but the difference still is less than 7%. Also, for the variable loads the difference is largest for short battery lifetimes, with a maximum of 5.4% for Case 21.

Figure 5 shows a plot of the lifetimes computed with both models versus the lifetimes computed with the electro-chemical simulation program DUALFOIL. In comparison with DUALFOIL both models overestimate the battery lifetime for the low continuous loads (long lifetimes), with errors growing upto 10%. The results of the variable loads are even better, with a maximum error of 5%.

Case	DUALFOIL (min)	Diffusion (min)	KiBaM (min)	Case	DUALFOIL (min)	Diffusion (min)	KiBaM (min)
C1	36.4	36.2	36.3	C12	159.0	155.4	154.1
C2	57.2	55.8	55.7	C13	133.8	131.7	131.3
C3	74.2	71.9	71.4	C14	132.9	129.7	129.4
C4	128.1	124.9	123.6	C15	207.6	209.2	209.2
C5	178.5	176.7	175.7	C16	202.4	200.7	200.7
C6	41.5	41.0	41.1	C17	253.8	251.2	250.8
C7	30.6	30.8	30.5	C18	204.6	204.6	204.3
C8	37.0	37.4	38.1	C19	209.4	208.7	208.2
C9	35.4	35.2	34.8	C20	31.7	33.2	31.5
C10	135.2	132.6	131.7	C21	55.9	55.9	58.8
C11	108.8	107.4	107.9	C22	97.5	94.5	94.3

Table 2: Lifetimes of variable-load profiles (cf. Appendix A) computed with DUALFOIL, the diffusion model (both from [13]) and KiBaM (computed by us)

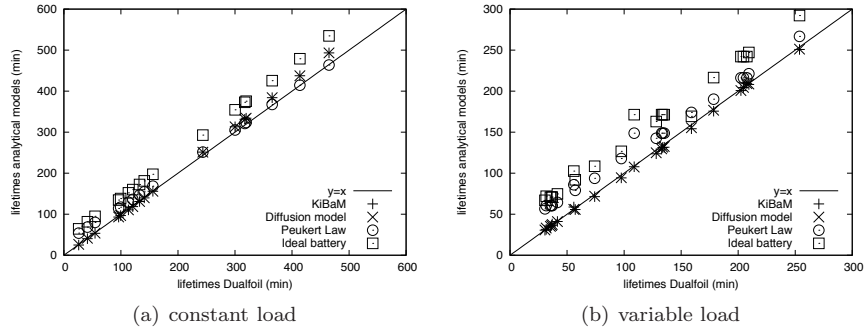


Figure 5: Computed lifetimes according to the Dualfoil simulation program versus the diffusion model and the KiBaM for constant loads (a) and variable loads (b). Next to the two analytical models, the lifetimes according to the formulas of the ideal battery (1) and Peukert's law (3) are shown.

Besides the results of the two models, also the lifetimes according to Peukert's law and the ideal battery model are shown in Figure 5. The ideal battery model always predicts longer lifetimes, since it does not take into account any loss of capacity due to the rate capacity effect. Also, Peukert's law overestimates the battery lifetimes for most cases. Only for the high continuous loads it gives better predictions than the KiBaM and diffusion model.

5 Limitations of analytical battery models

In the previous section we have seen that both models give nearly the same results. In this section, all further results are obtained with the KiBaM, but the conclusions also apply on the diffusion model.

With the KiBaM the effect of a varying load on the charge delivered by the battery was analysed in more detail. A square wave, switching between on (1 A) and off (0 A), was used as load. In Figure 6 the charge delivered is shown as a

function of the frequency of the periodic load. For low frequencies the delivered charge is constant, because the battery is emptied during the first on-period. Therefore, the charge delivered is equal to the case of continuous discharge at 1 A. When the frequency is increased, one sees a sudden discontinuous increase of the charge delivered by the battery. At the point of this jump, the battery is nearly empty at the end of the first on-period, and it has an off-period to recover some of its capacity. The recovered charge can be used in the next on-period, resulting in a considerable increase of the delivered charge. After this increase, the delivered charge slowly decreases when the frequency is further increased. The explanation of this decrease is twofold. First, the off-period is shorter and therefore there is less time for recovery. Second, the first on-period is shorter and less charge is delivered to the load during this time.

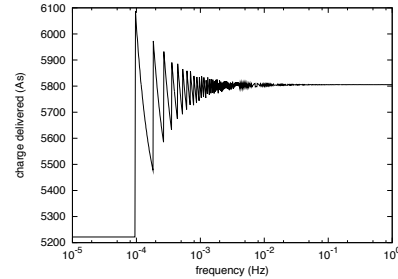


Figure 6: Charge delivered by the battery as a function of the frequency a square wave load. The charge delivered is computed using the KiBaM, with the parameters $c = 0.625$, $k = 4.5 \cdot 10^{-5} \text{min}^{-1}$ and the capacity of 7200 As.

Further increase of the frequency results in a discontinuous increase of the charge delivered each time the battery can recover during an extra off-period, followed again by a slow decrease. The increase gets smaller for higher frequencies since the extra recovery-time decreases. When the frequency is $> 10^{-2}$ Hz, the charge delivered is constant again. This is due to the short extra off-time, and the low frequency response at these high frequencies (cf. Section 4.2).

For the chosen load and set of battery parameters the charge delivered is highest for a frequency of $\sim 10^{-4}$ Hz. However, the position of the peaks depends highly on the battery parameters and the level of the on-current, and a slight variation might result in a big change in the charge delivered by the battery. In practice the battery parameters vary even between batteries of the same size and type. Therefore, it does not make sense to do battery lifetime predictions using single traces of a load profile. The used trace could result in a high performance of the battery with one set of the parameters, and a low performance with a slightly different set of parameters.

6 Conclusions & Outlook

The analysis of the KiBaM and diffusion model shows that the KiBaM is actually a first order approximation of the diffusion model. The parameters of the KiBaM can be adapted to make a better approximation of the diffusion model. The performed experiments with both models show that this approximation is very good for most practical loads. Therefore, it is better to use the more simple KiBaM model. However, one has to be careful using this type of model when drawing conclusions from only a few workload traces. A slight change in the battery parameters can change the battery lifetime dramatically especially when the load switching frequencies are low. A good way to avoid this problem is to

make use of stochastic workload models. With these models one can capture the full range of different possible workloadtraces. This results in a battery lifetime distribution, which tells us the probability of the battery being empty at time t given the type of workload. Comparing these probabilities one can find the best way to use the battery. Slight changes in the battery parameters, now, will not effect the results dramatically. One approach to do this is the by using Markov reward models, as described in [2]. Another approach is using prized timed automata [16] to describe the workload, and incorporate the Kinetic Battery Model into this model. This approach can help in finding the best scheduling scheme in a multi-battery system.

References

- [1] *IEEE Computer*, vol. 38, no. 11. IEEE Press, 2005.
- [2] L. Cloth, B. R. Haverkort, and M. R. Jongerden, “Computing battery lifetime distributions,” in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2007)*. IEEE Computer Society Press, 2007, pp. 780–789.
- [3] M. R. Jongerden and B. R. Haverkort, “Battery modeling,” Technical Report TR-CTIT-08-01, January 2008. [Online]. Available: <http://eprints.eemcs.utwente.nl/11645/>
- [4] M. Doyle, T. F. Fuller, and J. Newman, “Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell,” *Journal of the Electrochemical Society*, vol. 140, no. 6, pp. 1526 – 1533, 1993.
- [5] T. F. Fuller, M. Doyle, and J. Newman, “Simulation and optimization of the dual lithium ion insertion cell,” *Journal of the Electrochemical Society*, vol. 141, no. 1, pp. 1 – 10, 1994.
- [6] —, “Relaxation phenomena in lithium-ion-insertion cells,” *Journal of the Electrochemical Society*, vol. 141, no. 4, pp. 982 – 990, 1994.
- [7] S. C. Hageman, “Simple PSpice models let you simulate common battery types,” *Electronic Design News*, vol. 38, pp. 117 – 129, 1993.
- [8] J. Manwell and J. McGowan, “Lead acid battery storage model for hybrid energy systems,” *Solar Energy*, vol. 50, pp. 399–405, 1993.
- [9] —, “Extension of the kinetic battery model for wind/hybrid power systems,” in *Proceedings of the 5th European Wind Energy Association Conference (EWEC '94)*, 1994, pp. 284–289.
- [10] J. Manwell, J. McGowan, E. Baring-Gould, S. W., and A. Leotta, “Evaluation of battery models for wind/hybrid power system simulation,” in *Proceedings of the 5th European Wind Energy Association Conference (EWEC '94)*, 1994, pp. 1182–1187.
- [11] D. Rakhmatov and S. Vrudhula, “An analytical high-level battery model for use in energy management of portable electronic systems,” in *Proceedings of the International Conference on Computer Aided Design (ICCAD'01)*, 2001, pp. 488–493.

- [12] D. Rakhmatov, S. Vruthula, and D. A. Wallach, “Battery lifetime predictions for energy-aware computing,” in *Proceedings of the 2002 International Symposium on Low Power Electronics and Design (ISLPED '02)*, 2002, pp. 154–159.
- [13] —, “A model for battery lifetime analysis for organizing applications on a pocket computer,” *IEEE Transactions on VLSI Systems*, vol. 11, no. 6, pp. 1019–1030, 2003.
- [14] R. Rao, S. B. K. Vruthula, and N. Chang, “Battery optimization vs energy optimization: which to choose and when?” in *Proceedings of the International Conference on Computer Aided Design (ICCAD'05)*. IEEE Computer Society, 2005, pp. 439–445.
- [15] Fortran programs for the simulation of electrochemical systems. [Online]. Available: <http://www.cchem.berkeley.edu/jsngrp/fortran.html>
- [16] G. Behrmann, K. G. Larsen, and J. I. Rasmussen, “Optimal scheduling using priced timed automata,” *SIGMETRICS Performance Evaluation Review*, vol. 32, no. 4, pp. 34–40, 2005.

A Appendix

Case	Description	Timing (min)
C1	IAT-off-IAT	(0, 19.5, 26.0)
C2	IAR-off-IAR	(0, 31.0, 41.3)
C3	IST-off-IST	(0, 41.0, 54.6)
C4	ISR-off-ISR	(0, 74.6, 99.5)
C5	MPEG-off-MPEG	(0, 105.7, 140.9)
C6	IAT-off-IAT	(0, 19.5, 29.9)
C7	IAT-off-IAT	(0, 19.5, 22.1)
C8	IAT-off-IAT	(0, 23.4, 29.9)
C9	IAT-off-IAT	(0, 15.6, 22.1)
C10	Boot-IAT-IAR-MSD-DSD-TSD-WSD-IAD	(0, 0.5, 5.5, 10.5, 35.5, 60.5, 85.5, 110.5)
C11	Boot-WSD-TSD-DSD-MSD-IAR-IAT-IAD	(0, 0.5, 25.5, 50.5, 75.5, 100.5, 105.5, 110.5)
C12	Boot-WSD-TSD-DSD-MSD-IAR-off-... Boot-IAT-IAD	(0, 0.5, 25.5, 50.5, 75.5, 100.5, 105.5, ... 130.5, 131.0, 136.0)
C13	Boot-[IAT-IAR-MSD-DSD-TSD-WSD] ⁵ -IAD	(0, [0.5, 1.5, 2.5, 7.5, 12.5, 17.5] _{22.5} ⁵ , 110.5)
C14	Boot-[WSD-TSD-DSD-MSD-IAR-IAT] ⁵ -IAD	(0, [0.5, 5.5, 10.5, 15.5, 20.5, 21.5] _{22.5} ⁵ , 110.5)
C15	MPEG-Dictation-Talk1-WaV1-MPEG	(0, 50.0, 100.0, 150.0, 200.0)
C16	WAV1-Talk1-Dictation-MPEG-MPEG	(0, 50.0, 100.0, 150.0, 200.0)
C17	WAV1-Talk1-Dictation-off-MPEG-MPEG	(0, 50.0, 100.0, 150.0, 200.0, 250.0)
C18	[WAV1-Talk1-Dictation-MPEG] ¹⁰ -MPEG	([0, 5.0, 10.0, 15.0] _{20.0} ¹⁰ , 200)
C19	[WAV2-Talk3-Dictation-MPEG] ¹⁰ -MPEG	([0, 5.0, 10.0, 15.0] _{20.0} ¹⁰ , 200)
C20	[IAR-IAT] [∞]	([0, 1.0] _{2.0} [∞])
C21	[IAR-IAT-IST] [∞]	([0, 1.0, 2.0] _{3.0} [∞])
C22	5.0 + (5.0 per min)	(0, 1.0, 2.0, ...)

Table 3: The simulated variable-load profiles [13]

Analytical TCP Throughput Model for HSDPA*

Levente Bodrog[†] Gábor Horváth[‡] Csaba Vulkán[§]

Abstract

In this paper, an approximate, Padhye model based TCP throughput calculation method is presented for mobile data services over HSDPA. The Padhye model is defining the TCP throughput based on two input parameters: the packet loss probability and the TCP Round Trip Time. In order to provide the input parameters for the TCP throughput calculation, an equivalent queuing network model of the HSDPA system is created, which includes the congestion points and protocol layers that are having dominant impact on the delay and packet drop. The solution of the queuing network model is described in detail. Finally, the model is validated with NS2 simulations.

1 Introduction

HSDPA (High Speed Downlink Packet Access) provides a packet based downlink service for data users over the UMTS (Universal Mobile Telecommunications System) with data rates ranging up to several megabits per second. [7]

In conventional UMTS, the Layer 2 protocols of the radio protocol interface, such as Radio Link Control (RLC) and Medium Access Control (MAC) protocol are terminated in the Radio Network Controller (RNC). Physical layer protocols of the radio interface are implemented in the Node-B that is connected to the RLC via the Iub interface. In Acknowledged Mode (AM), the RLC is responsible for error-free, in-sequence delivery of the user data. This is achieved by retransmissions based on the (Automatic Repeat Request) ARQ mechanism. RLC retransmissions are increasing the Layer 2 Round Trip Time (RTT) and may trigger TCP timeouts.

HSDPA is introducing an additional protocol layer located in the Node-B (see Figure 1), namely MAC-hs, which makes possible Node-B controlled fast adaptation of the modulation and coding scheme, fast scheduling and retransmission handling with the Hybrid ARQ (HARQ) functionality. This solution is reducing the Layer 2 RTT when retransmissions are required due to erroneous data transfer. Although retransmissions are handled by the Node-B, RLC ARQ has not been removed from the system in order to be compatible with the

*The research work of Levente Bodrog and Gábor Horváth is partially supported by the Hungarian Research Found (OTKA) under grant K61709. The content of this paper has been developed in cooperation with Nokia Siemens Networks.

[†]Technical University of Budapest, Dept. of Telecommunications, bodrog@hit.bme.hu

[‡]Technical University of Budapest, Dept. of Telecommunications, ghorvath@hit.bme.hu

[§]Nokia Siemens Networks, Budapest, Hungary, csaba.vulkan@nsn.com

Rel'99 network solutions and to provide the capability of soft handover control. Retransmissions are still handled by RLC if the maximum allowed number of MAC-hs retransmissions is exceeded or there are packet drops on the transport network. The RLC retransmissions are increasing the round trip time of the data connections using HSDPA service. These factors and the in-sequence delivery of the user packets by the RLC are causing that the TCP flow control is not able to detect and resolve the congestion situation on the Iub interface. As a result, the TCP notices the congestion only upon the expiry of the Timeout timer or when finally the RLC discards the packets that has reached the maximum number of retransmissions.

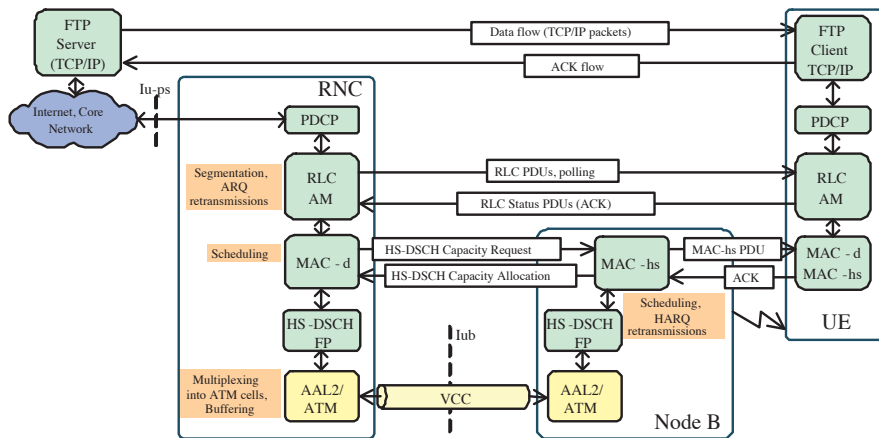


Figure 1: The overview of the protocols of HSDPA

The distribution of the radio protocol architecture between RNC and Node-B requires that a flow control algorithm – the HSDPA flow control [10] – is implemented. With this algorithm the Node-B controls the amount of data sent from the RNC in order to keep its buffers at optimal level so that the air interface capacity is not wasted, and in the same time the delay on the Node-B buffer is not too high. Typically, the HSDPA flow control is measuring the Node-B buffer size and the amount of transferred Packet Data Units (PDUs) over a sampling period without considering the available resources on the Iub transport network shared by Real Time (RT) Non Real Time (NRT) and HSDPA services.

A good indicator of the level of service an HSDPA access network can provide to the mobile users is the achievable TCP throughput. TCP performance in HSDPA has been considered in [3], [2] and [1] in the past. These papers are presenting a detailed simulation based analysis of the TCP behaviour over HSDPA systems, but the results are based on simulations. In this paper we propose an analytical throughput model for TCP connections over HSDPA.

The rest of the paper is organized as follows. Section 2 gives a short technological overview on the HSDPA UTRAN, describes how the packets are delivered from the RNC to the User Equipments (UE) and introduces a queueing network model of the system. Section 3 summarizes the concept of the approximate throughput calculation and describes in detail the solution of the queueing network model of the system. A numerical example is provided in Section 4,

finally Section 5 concludes the paper.

2 System overview and the equivalent queuing model

The overview of the radio access network configuration in case of HSDPA service is shown in Figure 1. After header compression in the Packet Data Convergence Protocol (PDCP) layer, the incoming data (TCP/IP) packets are segmented and encapsulated by the RLC AM entity. These segments (PDUs) are scheduled by the MAC-d layer according to the HSDPA flow control commands. The RLC entity is actively polling the User Equipment (UE) that is responding with Status PDUs indicating the sequence number of lost and received PDUs. Lost PDUs are retransmitted. The master of the HSDPA flow control is the MAC-hs located in the Node-B. It grants resources to the HSDPA connections (MAC-d flows) at RNC by sending High Speed Dedicated Shared Channel (HS-DSCH) Capacity Allocation message that includes the allocation size i.e. the number of PDUs and their maximum size (HS-DSCH Credits, MAC-d PDU Length); the interval the data can be sent at (HS-DSCH Interval), and the validity period of the allocation (HS-DSCH Repetition Period). This message is sent either solicited, upon reception of a HS-DSCH CAPACITY REQUEST message from the RNC, or unsolicited. The HS-DSCH Frame Protocol (FP) assembles a frame out of the scheduled PDUs as it is specified in and transfers it to the ATM Adaptation Layer type 2 (AAL2), where these frames are segmented to 45 bytes, and encapsulated into Common Part Sublayer (CPS) packets. The size of the CPS-Packet header is 3 byte thus the maximum size of one packet is 48 byte. The CPS-Packets are eventually assembled into CPS PDUs and sent to the destination via the Virtual Channel Connection (VCC). The CPS-PDU header is one byte long thus at maximum 47 CPS-Packet bytes can be fitted into one Asynchronous Transfer Mode (ATM) cell. As queues are intrinsic to the HSDPA system, a natural approach to model the TCP RTT (Round Trip Time) – that is an important parameter with impact on the overall TCP performance – is to create an equivalent queuing model. Accordingly, the potential bottleneck points that are dominating the downlink delay has to be identified (in case of mobile services the users are mainly downloading content to their mobiles causing loaded system in downlink). The developed model focuses on the downlink performance, whereas the uplink delay is modelled with a constant delay. Packets drop (p) can appear at these bottleneck points due to buffer overflow or when the maximum number of retransmissions is reached. There are three such points in the system:

- The buffers of the RLC layer where the RLC PDUs (resulted from the segmentation of the user packets) are stored until a positive acknowledgement arrives or the maximum number of retransmissions is reached and the RLC AM entity discards them. The RLC buffers are scheduled by the MAC-d layer based on the credits received from the Node-B (MAC-hs entity). These credits are calculated in order to maximize the air interface throughput not necessarily taking into consideration the congestion situation over the Iub links thus the RLC layer can easily overload the transport network. In this model it is assumed that the uplink delay of

the HS-DSCH CAPACITY ALLOCATION message is zero.

- The buffers of the AAL2/ATM transport network. As the transport network is a shared and limited resource, congestion may occur leading to increased delay and eventually to packet drops. In this paper the transport network is modelled with one buffer corresponding to the bottleneck link.
- The MAC-hs buffers in Node-B. There is one buffer per each MAC-d flow (HSDPA connection) that is storing the MAC-d PDUs waiting for transmission. The amount of the MAC-d PDUs that can be transmitted in a 2ms long TTI depends on the reported channel quality indicator (CQI). In case of transmission failure, the MAC-d PDUs are retransmitted. If the maximum number of retransmissions is reached the MAC-d PDUs are discarded by the HARQ and the RLC ARQ will handle the retransmissions.

An overview of the queueing network model of the system is shown on Figure 2. The three components of the queueing model i.e. the RLC buffers, the transport buffers and the MAC-hs buffers are located at different protocol layers. Each flow has a dedicated buffer at the RLC layer that stores the PDUs resulted from the segmentation of the TCP packets. The MAC-d schedules these buffers independently based on the credits received from the Node-B. PDUs are discarded in case of buffer overflow or when the maximum number of retransmissions is reached. Each PDU is stored in the buffer until the positive acknowledgement is received or when PDU Discard procedure is executed by the RLC.

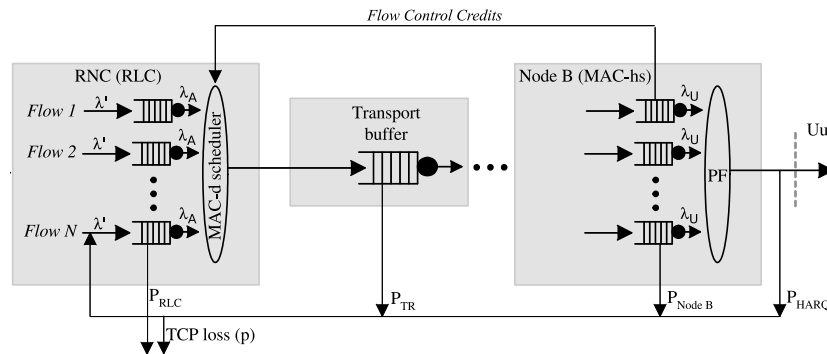


Figure 2: The overview of the queueing network model of the system

The transport network is modelled by one buffer representing the bottleneck link. ATM cells are discarded at buffer overflow. At the Node-B, each MAC-d flow has a dedicated buffer. At each 2 ms TTI the Proportional Fair (PF) scheduler is selecting the buffer to be served based on the average throughput of each flow and their instantaneous channel quality. Upon an erroneous transmission over the air interface, the PDUs are retransmitted until the maximum number of transmissions is reached.

3 The concept of the TCP throughput calculation

There are several models available to calculate the TCP throughput. The most popular one is the so-called Padhye model [11]. This model essentially gives a simple formula that expresses the TCP throughput (B) as a function of the packet loss (p) and round trip time (RTT)

$$B(p, RTT) = \begin{cases} \frac{\frac{1-p}{p} + E(W) + \hat{Q}(E(W)) \frac{1}{1-p}}{RTT \left(\frac{b}{2} E(W_u) + 1 \right) + \hat{Q}(E(W)) T_0 \frac{f(p)}{1-p}}, & \text{if } E(W_u) < W_{\max} \\ \frac{\frac{1-p}{p} + W_{\max} + \hat{Q}(E(W)) \frac{1}{1-p}}{RTT \left(\frac{b}{8} W_{\max} + \frac{1-p}{p W_{\max}} + 2 \right) + \hat{Q}(W_{\max}) T_0 \frac{f(p)}{1-p}}, & \text{otherwise.} \end{cases} \quad (1)$$

In the formula p denotes the packet loss probability, b is the number of packets covered by one acknowledgement ($b = 1$ is assumed in this paper), T_0 is the timeout (we use $T_0 = 1.5$ sec), RTT is the Round Trip Time of the packets, W_{\max} is the maximum Congestion Window size, $E(W_u)$ is the mean Unconstrained Window size given by:

$$E(W_u) = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}.$$

$\hat{Q}(w)$ is the probability that a loss in a window of size w is due to Timeout, calculated with the formula:

$$\hat{Q}(w) = \min \left(1, \frac{\left(1 - (1-p)^3\right) \left(1 + (1-p)^3 \left(1 - (1-p)^{w-3}\right)\right)}{1 - (1-p)^w} \right).$$

Finally, $f(p)$ is a simplifying notation:

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6.$$

Thus, the two unknown parameters of the TCP throughput calculation are the Round Trip Time (RTT) and the packet loss probability (p). Since the major part of the RTT is spent as waiting time in the buffers of the network devices, and the packet loss occurs due to saturated buffers or air interface errors, we model the HSDPA system by a queueing network. To reduce complexity, we decided not to involve the micro-behaviour of the TCP flow control into the model. Instead, we consider the TCP traffic as a flow of packets having a constant intensity.

By assuming a constant rate TCP traffic, the RTT and p are calculated using the queueing network model of the system described in detail in the next sections. Once the RTT and p are known, the TCP traffic intensity corresponding to p and RTT can be calculated with the Padhye model. This value is not necessarily equal to the TCP rate assumed initially. In this case the initially assumed TCP rate is adjusted, and throughput calculation is repeated until the equilibrium is reached. The output of the method will be the TCP rate B^* that – when loaded into the queueing network model – results in a p and RTT such that the Padhye model provides the same TCP throughput, thus $B^* = B(p, RTT)$.

3.1 Overview of the calculation algorithm

As described in Section 3, the TCP throughput over HSDPA is calculated as the load (λ_{TCP}) that carried over the network causes a Round Trip Time RTT and packet loss p such that the Padhye formula results in the same throughput i.e. $B(p, RTT) = \lambda_{\text{TCP}}$.

This equilibrium is reached by a simple interval bisection method summarized in Algorithm 1. At the beginning of the algorithm, the lowest possible throughput is initialized to $a_1 = 0$. The mean TCP throughput can not be larger than the average air interface throughput, thus the upper limit of the interval bisection can be initialized to $E(S_{\text{Node-B}})$. In each step, the queueing network shown on Figure 2 is analysed, the packet loss and mean RTT is calculated. The upper and lower bounds of the interval are adjusted depending on the relation of the actual TCP throughput assumption, λ_{TCP} and the throughput resulted by the Padhye formula $\lambda_{\text{PADHYE}} = B(p, RTT)$.

Algorithm 1 The TCP Throughput Calculation Algorithm

INPUT: `sysparam`//the system parameters are listed in Table 1

OUTPUT: γ //the TCP throughput

```

1:  $a_1 = 0$ //the lowest possible throughput value
2:  $a_2 = E(S_{\text{Node-B}})$ //the upper bound is the average air interface throughput
3: while  $|\lambda_{\text{TCP}} - \lambda_{\text{old}}| > \epsilon$  do //the loop of the bisection method
4:    $\lambda_{\text{TCP}} = \frac{a_1 + a_2}{2}$ 
5:    $p, RTT = \text{QN\_Analysis}(\lambda_{\text{TCP}})$ 
6:    $\lambda_{\text{PADHYE}} = B(p, RTT) \cdot K f_T / f_M$  //Apply Padhye model as in (1)
7:   if  $\lambda_{\text{PADHYE}} > \lambda_{\text{TCP}}$  then
8:      $a_1 = \lambda_{\text{TCP}}$ 
9:   else
10:     $a_2 = \lambda_{\text{TCP}}$ 
11:    $\lambda_{\text{old}} = \lambda_{\text{TCP}}$ 
12: return  $\gamma = \lambda f_M$  //harmonize units

```

The parameters p and RTT are calculated by the analysis of the queueing network (Figure 2). The users are assumed to be identical, the calculation is performed for one selected (tagged) user. Accordingly the queueing network seen by the tagged user consists of three nodes: the RLC buffer, the transport (ATM) buffer and the MAC-d buffer at the Node-B. This queueing network does not belong into the class of queueing networks for which exact solution is known, thus a traffic decomposition based approximate analysis has been developed. [4] The analysis starts with the first queue. In addition to the performance measures of interest, the output process has to be approximated, too. This approximate departure process is the arrival process to the next queue in the network, that can be analysed in the same way. The calculation is repeated until the last queueing node is analysed. As the network has feed back traffic (RLC loss is modelled as if the lost PDUs would be re-inserted into the RLC buffer), an iterative solution method has to be used. Initially it is assumed that there is no feed back traffic; the whole network is analysed and the feed back traffic (amount of PDUs that must be retransmitted) is calculated. In the next iteration step this feed back traffic is taken into consideration during the analysis of the first queue thus in the whole queueing network model. The iterations are repeated

Description	notation	value
The number of HSDPA users	K	16
The RLC buffer size [PDUs]		1000
Transport node buffer size [ATM cells]	L	2000
Node-B buffer size [PDUs]		1000
Maximum number of RLC (re)transmissions	R	6
Maximum number of HARQ (re)transmissions	M	3
TCP packets acknowledged by one ACK	b	1
TCP Timeout interval	T_0	1.5 sec
Maximum TCP Congestion Window size	W_{\max}	48 KB
Block Error Rate over the air interface	P_e	0.01
Prob. of two successive erroneous transmissions	P_s	0.001
Service distribution at the air interface	$\Pr(\hat{S} = k)$	from trace file
TCP packet size	f_T	1500 byte
Size of MAC-d and RLC PDUs	f_M	336 bit
Accuracy parameters	ϵ, ϵ'	1
Transport link capacity	C	

Table 1: The parameters contained in `sysparam`**Algorithm 2** $\lambda_{\text{out}} = \text{QN Analysis}(\lambda_{\text{in}})$ **INPUT:** λ_{in} //the load generated by the TCP sources**OUTPUT:** p, RTT //packet loss and mean round trip time

- 1: $\lambda' = \frac{\lambda_{\text{in}}}{K}$ //the throughput of the tagged HSDPA user
- 2: **while** $|\lambda' - \lambda'_{\text{old}}| > \epsilon'$ **do** //loop to find the equilibrium value of λ'
- 3: $(P_{\text{RLC}}, E(T_{\text{RLC}}), D_{\text{RLC}}) = \text{solve_rlc}(\lambda')$ //see Section 3.2
- 4: $(P_{\text{Tr}}, E(T_{\text{Tr}}), D_{\text{Tr}}) = \text{solve_tr}(C, D_{\text{RLC}})$ //see Section 3.3
- 5: $(P_{\text{Node-B}}, E(T_{\text{Node-B}}), \lambda_U) = \text{solve_node-b}(S, D_{\text{Tr}})$ //see Section 3.4
- 6: $p_L \leftarrow (P_{\text{Tr}}, P_{\text{Node-B}}, P_{\text{HARQ}})$ //the loopback probability given by (15)
- 7: $\hat{p} = \frac{\sum_{k=1}^R \frac{(1-p_L)^{k-1} p_L}{\sum_{k=1}^R \frac{(1-p_L)^{k-1} p_L}{1-p_L^R}}}{\sum_{k=1}^R \frac{(1-p_L)^{k-1} p_L}{1-p_L^R}}$ //the probability that the PDU is resent by RLC
- 8: $\lambda' = \frac{\lambda_{\text{in}}}{K} + \hat{p} \cdot p_L \cdot \lambda_A$
- 9: $\lambda'_{\text{old}} = \lambda'$
- 10: $(D_u, D_s) \leftarrow (P_{\text{Tr}}, P_{\text{Node-B}}, P_{\text{HARQ}}, E(T_{\text{RLC}}), E(T_{\text{Tr}}), E(T_{\text{Node-B}}))$ //(19)
- 11: $RTT = \sum_{k=1}^R \frac{p_L^{k-1} (1-p_L)}{1-p_L^R} ((k-1) D_u + D_s)$ //the RTT derived in (20)
- 12: $p = 1 - \frac{\lambda_U}{\lambda}$ //the TCP loss probability given in (21)

until the difference in the results will not exceed the accuracy parameter. The algorithm is summarized on Algorithm 2.

3.2 The model of the RLC buffer

The model of the RLC layer (referred to as `solve rlc` in line 3 in Algorithm 2) is based on the observation that the *service process* of the RLC buffer (thus, the arrival process of the RLC PDUs to the transport network) is controlled by the HSDPA flow control. To achieve efficient air interface resource usage, the Node-B grants credits to each flow based on the reported channel quality and the measured average throughput of the flows. At each HSDPA scheduling interval the MAC-d scheduler will transmit the amount of PDUs defined by the received credits. In this paper we assume that scheduling interval is 10 ms (that is a typical value), thus PDUs are scheduled at each $TTI_{\text{RLC}} = 10$ ms. The calculation of the amount of PDUs that can be sent at a given scheduling instance is based on the assumption that the Node-B has a perfect knowledge on the air interface conditions, thus that the distribution of the number of MAC-d PDUs that can be transmitted over the air-interface at each 2 ms HSDPA TTI is known. (See Section 3.4).

Based on this assumption the number of PDUs the MAC-d scheduler is transferring during a 10 ms time slot (denoted by S_{RLC}) is given by:

$$S_{\text{RLC}} = \sum_1^5 S_{\text{Node-B}}.$$

The *arrival process* to the RLC buffer consists of the incoming traffic to the system (having an intensity of λ_{in}/K) and the PDUs that are retransmitted by the RLC AM entity (this is how RLC losses, denoted by λ_{FB} are modelled). λ_{FB} is calculated with the equation (17). At this calculation step Poisson traffic with a total arrival rate of λ' is assumed:

$$\lambda' = \frac{\lambda_{\text{in}}}{K} + \lambda_{\text{FB}}.$$

The distribution of the number of packets entering arriving to the RLC buffer in a 10 ms interval is calculated as follows:

$$\Pr(A_{\text{RLC}} = k) = \frac{(\lambda' TTI_{\text{RLC}})^k}{k!} e^{-\lambda' TTI_{\text{RLC}}} \quad k = 0, 1, 2, \dots \quad (2)$$

The distribution is truncated at N such that the probability of the cut-off part of the distribution is reasonably small.

The *queue length* evolution embedded at TTI_{RLC} long time slots is then modelled by a discrete time Markov chain (DTMC) according to the following evolution equation:

$$X_{n+1} = (X_n + A_{n+1} - S_{n+1})^+,$$

where X_{n+1} is the queue length, A_{n+1} is the number of arrivals and S_{n+1} is the number of packets served in the $n + 1$ st time slot. $(\cdot)^+$ denotes $\max(0, \cdot)$.

The ij th element of the transition probability matrix (\mathbf{P}) of the DTMC is given by:

$$p_{ij} = \begin{cases} \sum_{k=0}^{\infty} \Pr(A_{\text{RLC}} = k) \Pr(S_{\text{RLC}} = j - i + k) & i < L - N \\ \sum_{k=0}^{L-i} \Pr(A_{\text{RLC}} = k) \Pr(S_{\text{RLC}} = i - j + k) + \\ + \Pr(S_{\text{RLC}} = L + 1 - j) \sum_{k=L-i+1}^N \Pr(A_{\text{RLC}} = k) & i \geq L - N. \end{cases}$$

L is the length of the RLC buffer, N is the support of the arrival distribution. In the first case the queue level is as low that no loss can happen, thus the transition probability equals the probability that there were $j - i$ more packets served than arrived. In the second case, the first (second) term corresponds to arrival sizes without (with) loss, respectively.

The steady state solution ($\boldsymbol{\pi}$) of the DTMC is given by the solution of the linear equation system

$$\begin{aligned} \boldsymbol{\pi} \mathbf{P} &= \boldsymbol{\pi} \\ \boldsymbol{\pi} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} &= 1. \end{aligned}$$

Having the steady state solution, the loss probability at the RLC buffer is calculated as the ratio of the mean number of lost and of the mean number of arrived PDUs during a $TTI_{\text{RLC}} = 10$ ms time slot:

$$P_{\text{RLC}} = \frac{\sum_{i=0}^L \pi_i \sum_{j=0}^N \max(0, i + j - L) \Pr(A_{\text{RLC}} = j)}{\sum_{i=0}^L \pi_i \sum_{j=0}^N j \Pr(A_{\text{RLC}} = j)}. \quad (3)$$

The system time of the PDUs in the RLC buffer is calculated using Little's theorem:

$$E(T_{\text{RLC}}) = \frac{E(X_{\text{RLC}})}{(1 - P_{\text{RLC}}) E(A_{\text{RLC}})} TTI_{\text{RLC}} + \frac{1}{2} TTI_{\text{RLC}}, \quad (4)$$

where $E(X_{\text{RLC}})$ is the mean queue length. Since this is a discrete time model but arrivals can happen in continuous time, the model does not differentiate between arrivals at the beginning of the scheduling interval and at the end of it i.e. as they would not have different system times. Assuming that the arrival instants are uniformly distributed over the scheduling interval, the system time computed from the embedded DTMC is increased with the half of the interval.

During the analysis of the queueing network, the *departure process* from the RLC buffers has to be calculated as this is the arrival process to the transport

buffer. We assume that the departures are independent and identically distributed (i.i.d.), with the distribution of the number of departing packets in a TTI_{RLC} interval computed by:

$$\begin{aligned} \Pr(D_{\text{RLC}} = k) = & \sum_{i=0}^L \pi_i \sum_{j=k+1-i}^{\infty} \Pr(A_{\text{RLC}} = j) \Pr(S_{\text{RLC}} = k) + \\ & + \sum_{i=0}^L \pi_i \Pr(A_{\text{RLC}} = k - i) \sum_{j=k}^{\infty} \Pr(S_{\text{RLC}} = j). \end{aligned} \quad (5)$$

This expression consist of two terms: the first corresponds to the case when there are enough packets in the buffer, the number of departing packets is determined by the number of packets the server can serve whereas in the second term the server could serve more packets than the buffer content.

3.3 The model of the transport buffer

In this paper we consider an AAL2/ATM based transport network (the transport link buffer model and its solution is referred in line 4 of Algorithm 2). The AAL2 layer is multiplexing the user connections into one Constant Bit Rate (CBR) VCC, with capacity C .

The ATM switch works in continuous time in contrast with the MAC-d and PF schedulers that are working in time slotted manner. In order to avoid mixing the continuous and discrete models, we decided to apply a discrete time model for the transport buffer as well. The RLC buffer is scheduled with $TTI_{\text{RLC}} = 10$ ms transmission interval and the PF scheduler in Node-B is forwarding PDUs with a $TTI_{\text{Node-B}} = 2$ ms. The selected time slot for the transport buffer is the minimum of these two e.g. $TTI_{\text{Tr}} = 2$ ms is used to approximate the transport buffer mainly because this value allows finer resolution in time than a model with 10 ms interval. Another assumption is that in the model the transport buffer stores and transmits RLC PDUs instead of ATM cells. Since the RLC PDUs are the “data units” in other parts of the network, using the same data unit in the transport buffer simplifies the calculation significantly.

The distribution of the number of *arrivals* in a time slot is derived from the distribution of the number of departures from the RLC layer (D_{RLC}). The departure process of the RLC corresponds to a 10 ms TTI_{RLC} , while the transport buffer model has a 2 ms TTI_{Tr} . Thus, as a first step a conversion has to be applied between the MAC-d scheduling interval and the transport time slot, having a departure distribution from the RLC layer in a five-times longer TTI_{RLC} . The following binomial assumption is applied:

$$\Pr(D^{\text{ms}_{\text{tr}}} = k) = \sum_{i=k}^{\infty} \Pr(D_{\text{RLC}} = i) \binom{i}{k} \left(\frac{1}{5}\right)^k \left(1 - \frac{1}{5}\right)^{i-k},$$

where $\Pr(D^{\text{ms}_{\text{tr}}} = k)$ is the probability of k arriving packets in a TTI_{Tr} time period if there were i arrivals in the TTI_{RLC} time period or in other words to choose k arrivals from i with probability $\frac{1}{5}$ – the quotient of the lengths of the two kinds of TTIs.

When calculating the distribution of the number of arrivals to the transport buffer, the whole traffic aggregate has to be considered each user connection is

multiplexed into one VCC:

$$A_{\text{Tr}} = \sum_1^K D^{2\text{ms}_{\text{tr}}},$$

where K is the number of HSDPA users.

The *service time* of the RLC PDUs in the transport buffer is calculated as

$$D = \frac{\text{RLC packet size with overheads}}{C}.$$

The transport overheads are considered with following formula:

$$\text{RLC packet size with overheads} = f_M \cdot \underbrace{\left(\frac{53}{47} \frac{f_M + 24}{f_M} \frac{E(D_{\text{RLC}}) f_M + 72}{E(D_{\text{RLC}}) f_M} \right)}_{\text{overhead}}.$$

The overhead consists of the ATM header (40 bits) plus the 8 bit long CPS PDU Start Field (53/47), the 24 bit long CPS Packet header per an RLC PDU ($\frac{f_M + 24}{f_M}$) and finally the 72 bit long HS-DSCH FP frame header that carries $E(D_{\text{RLC}})$ RLC packets in an average.

In our discrete system having TTI_{Tr} long time slots the number of PDUs served in a time slot can either be $F = \lfloor \frac{TTI_{\text{Tr}}}{D} \rfloor$ or $F + 1$, according to the following probabilities:

$$\begin{aligned} \Pr(S_{\text{Tr}} = F) &= 1 - \left(\frac{TTI_{\text{Tr}}}{D} - F \right) \\ \Pr(S_{\text{Tr}} = F + 1) &= \frac{TTI_{\text{Tr}}}{D} - F. \end{aligned}$$

The *queue length* can be modelled by a DTMC similar to the one we applied for the RLC buffer, i.e.,

$$X_{n+1} = (X_n + A_{n+1} - S_{n+1})^+, \quad (6)$$

where X_{n+1} is the queue length, A_{n+1} is the number of arrivals and S_{n+1} is the number of PDUs served in the $n + 1$ st time slot.

Based on the distribution of the number of arrivals and served PDUs we can create the transition probability matrix of the DTMC such that the ij th element will be calculated in the same way as in the case of the RLC buffer:

$$p_{ij} = \begin{cases} \sum_{k=0}^{\infty} \Pr(A_{\text{Tr}} = k) \Pr(S_{\text{Tr}} = j - i + k) & i < L - (N - F) \\ \sum_{k=0}^{L-i} \Pr(A_{\text{Tr}} = k) \Pr(S_{\text{Tr}} = i - j + k) + \\ + \Pr(S_{\text{Tr}} = L + 1 - j) \sum_{k=L-i+1}^N \Pr(A_{\text{Tr}} = k) & i \geq L - (N - F). \end{cases}$$

The computation of the loss probability is similar to the one applied at the RLC modeling:

$$P_{\text{Tr}} = \frac{\sum_{i=0}^{L-F} \pi_i \sum_{j=0}^N \max(0, i+j-L) \Pr(A_{\text{Tr}} = j)}{\sum_{i=0}^{L-F} \pi_i \sum_{j=0}^N j \Pr(A_{\text{Tr}} = j)}. \quad (7)$$

The numerator is the expected number of lost PDUs and the denominator is the expected number of PDUs received correctly.

The system time of the PDUs in the transport buffer is calculated based on Little's theorem as (see (4)):

$$E(T_{\text{Tr}}) = \frac{E(X_{\text{Tr}})}{(1 - P_{\text{Tr}}) E(A_{\text{Tr}})} TTI_{\text{Tr}} + \frac{1}{2} TTI_{\text{Tr}}. \quad (8)$$

The departure process is calculated similar to the calculation of the same parameter in case of the RLC buffer

$$\begin{aligned} \Pr(D_{\text{Tr}} = k) &= \sum_{i=0}^{L-F} \pi_i \sum_{j=k+1-i}^{\infty} \Pr(A_{\text{Tr}} = j) \Pr(S_{\text{Tr}} = k) + \\ &+ \sum_{i=0}^{L-F} \pi_i \Pr(A_{\text{Tr}} = k-i) \sum_{j=k}^{\infty} \Pr(S_{\text{Tr}} = j). \end{aligned} \quad (9)$$

3.4 The Model of the MAC-hs Buffers

In this paper it is assumed that the MAC-hs buffers are scheduled by a Proportional Fair algorithm, that is making the scheduling decisions based on the instantaneous channel quality and the average throughput of the users with the scope to achieve high level of resource usage and in the same time to provide high level of fairness to the users. The scheduler is selecting one user for transmission at each scheduling instance (at every $TTI_{\text{Node-B}} = 2 \text{ ms}$). The reported Channel Quality Indicator (CQI) is defining the modulation and coding scheme, thus the number of MAC-d PDUs that can be transmitted during a TTI. Since the channel conditions can change quickly, temporary traffic overload can occur in the Node-B. The arriving PDUs are stored in the MAC-hs buffers (there is a separate buffer for each flow).

The modeling of the HSDPA air interface model is out of the scope of this paper. Instead, the MATLAB based tool of the Eurane project (see [6]) has been used in order to obtain the distribution of the number of MAC-d PDUs that can be transmitted in a TTI ($P(\hat{S} = k)$). This distribution has been generated by assuming saturated buffers without taking the impact of HARQ into consideration [9].

To obtain the *service process* of the MAC-hs buffer first the effect of HARQ is included in the model. According to [1,2] the probability of properly decoding the packet at the user side and thus the probability of the error free transmission after j trials is as follows:

$$P_j = \begin{cases} 1 - P_e, & j = 1 \\ P_e^{j-1} P_s^{j-2} (1 - P_e P_s), & j > 1. \end{cases}$$

The meaning and the default values of P_e and P_s are listed in Table 1. Considering that the maximal number of trials is M the expected number of retransmissions till success is as follows:

$$E(H) = \sum_{j=1}^M jP_j + M \left(1 - \sum_{j=1}^M P_j \right),$$

thus the probability that the time slot is lost due to a HARQ loss is:

$$P_{tl} = 1 - \frac{1}{E(H)}.$$

Finally, the distribution of the number of MAC-d PDUs that can be transmitted in a TTI taking the HARQ losses also into consideration is:

$$\Pr(S_{\text{Node-B}} = k) = \begin{cases} (1 - P_{tl}) \Pr(\hat{S} = k) + P_{tl} & k = 0, \\ (1 - P_{tl}) \Pr(\hat{S} = k) & k \neq 0. \end{cases} \quad (10)$$

The distribution of the number of *arrivals* to the MAC-hs buffer is calculated by assuming that the packets arriving from the transport network are directed to the buffer of the tagged user according to a random choice with probability $1/K$; resulting in the following binomial distribution:

$$\Pr(A_{\text{Node-B}} = k) = \sum_{i=k}^{\infty} \Pr(D_{\text{Tr}} = i) \binom{i}{k} \left(\frac{1}{K}\right)^k \left(1 - \frac{1}{K}\right)^{i-k}.$$

Contrary to the other two nodes the *queue length* evolution of the MAC-hs buffer is

$$X_{n+1} = (X_n - S_{n+1})^+ + A_{n+1},$$

This means that only those MAC-d PDUs can be served by the PF scheduler that have arrived before the beginning of TTI. The ij th element of the transition probability matrix is

$$p_{ij} = \begin{cases} \sum_{k=0}^{j-1} \Pr(A_{\text{Node-B}} = k) \Pr(S_{\text{Node-B}} = i - j + k) + \\ \quad + \Pr(A_{\text{Node-B}} = j) \sum_{k=i}^{\infty} \Pr(S_{\text{Node-B}} = k) & i < k_m \\ \sum_{k=0}^{\infty} \Pr(A_{\text{Node-B}} = k) \Pr(S_{\text{Node-B}} = j - i + k) & i \geq k_m. \end{cases}$$

After the computation of the steady state solution, the loss probability is calculated as the ratio of the lost and arrived PDUs in a TTI as:

$$P_{\text{Node-B}} = \frac{\sum_{i=0}^L \pi_i \sum_{j=0}^{\infty} \max(0, i + j - L) \Pr(A_{\text{Node-B}} = j)}{\sum_{i=0}^L \pi_i \sum_{j=0}^{\infty} j \Pr(A_{\text{Node-B}} = j)}. \quad (11)$$

$P_{\text{Node-B}}$ is the loss probability of PDUs due to buffer overflow and tail drop at the Node-B. However, at the Node-B the buffer overflow is not the only event that leads to packet loss. If the air interface quality is bad, and the HARQ mechanism fails, the MAC-hs discards the PDU from the corresponding HARQ register and the retransmission of the PDUs falls back to the RLC layer if the maximal number of retransmissions (M) has been reached. The probability of such events is denoted by P_{HARQ} and computed as:

$$P_{\text{HARQ}} = 1 - \sum_{j=1}^M P_j. \quad (12)$$

The system time of the MAC-hs buffer is calculated using Little's theorem as

$$E(T_{\text{Node-B}}) = \frac{E(X_{\text{Node-B}})}{(1 - P_{\text{Node-B}}) E(A_{\text{Node-B}})} TTI_{\text{Node-B}} + \frac{1}{2} TTI_{\text{Node-B}}, \quad (13)$$

where $E(X)$ is the mean queue length, and the addition of the extra time of half- $TTI_{\text{Node-B}}$ in the second term has the same explanation as in case of the RLC and transport network models.

For the queueing network analysis the departure intensity of the Node-B buffer is needed. The number of MAC-d PDUs per $TTI_{\text{Node-B}}$ equals the minimum of the number of packets in the buffer and the number of packets that can be served. This gives:

$$\lambda_U = \frac{1}{TTI_{\text{Node-B}}} \sum_{i=0}^L \pi_i \sum_{k=0}^{\infty} \Pr(S_{\text{Node-B}} = k) \min(i, k). \quad (14)$$

3.5 The Feed-back Link

In our queueing model the PDUs lost at the different parts of the network are considered as they were entering the RLC buffer again for repeated transmission. The feed-back link on Figure 2 “collects” these lost packets. In this section we calculate the traffic intensity on the feed-back link. This traffic (with Poisson assumption [4]) is added to the traffic entering the network during the analysis of the RLC model.

As a first step the probability of a PDU loss (due to any reason) in the network after leaving the RLC buffer is calculated. This probability is denoted by p_L and computed by:

$$p_L = P_{\text{Tr}} + (1 - P_{\text{Tr}}) P_{\text{Node-B}} + (1 - P_{\text{Tr}}) (1 - P_{\text{Node-B}}) P_{\text{HARQ}}. \quad (15)$$

It can happen that a retransmitted PDU is lost. After a given number of RLC level retransmission attempts (R) that equals the maximum number of RLC retransmissions the PDU is discarded and loss is detected by the TCP flow control. In this case this PDU does not enter the RLC buffer again (as long as the higher layer entity does not re-send it). The probability that a PDU loss did not reach the maximal number of retransmission attempts thus it increases the load of the RLC buffer is calculated with:

$$\hat{p} = \frac{\sum_{k=1}^R (1 - p_L)^{k-1} p_L}{\sum_{k=1}^{R+1} (1 - p_L)^{k-1} p_L} \quad (16)$$

(we assumed truncated geometrical distribution for the distribution of the number of retransmissions).

With the above considerations the traffic of the feed-back link is computed by:

$$\lambda_{FB} = \hat{p} \cdot p_L \cdot \lambda_A, \quad (17)$$

where λ_A denotes the mean departure rate of the RLC buffer.

3.6 The TCP Level Packet Loss and the RTT

In this section we describe the calculation of the TCP level performance measures based on the buffer-wise performance measures (given by equations (12), (3), (4), (7), (8), (11) and (13)).

The delay of one packet assuming that it has not been lost in the system is given by:

$$D_s = E(T_{RLC}) + E(T_{Tr}) + E(T_{Node-B}). \quad (18)$$

If it has been lost somewhere, the mean delay can be computed by:

$$\begin{aligned} D_u = & P_{Tr} E(T_{RLC}) + (1 - P_{Tr}) P_{Node-B} (E(T_{RLC}) + E(T_{Tr})) + \\ & + (1 - P_{Tr}) (1 - P_{Node-B}) P_{HARQ} \cdot \\ & \cdot (E(T_{RLC}) + E(T_{Tr}) + E(T_{Node-B})). \end{aligned} \quad (19)$$

If a packet has been retransmitted k times till successful transmission, the mean round trip time can be calculated as the sum of the mean delays of $k - 1$ unsuccessful transmissions and one times the delay of a successful transmission. Using the geometric distribution assumption for the number of retransmission attempts again we have

$$RTT = D_{UL} + \sum_{k=1}^R \frac{p_L^{k-1} (1 - p_L)}{1 - p_L^R} ((k - 1) D_u + D_s), \quad (20)$$

where D_{UL} denotes the mean delay in uplink direction considered to be constant as the UTRAN is typically not congested in uplink direction.

The loss at TCP layer is simply calculated by one minus the ratio of the traffic entering and leaving the system:

$$p = 1 - \frac{\lambda}{\lambda_U} \quad (21)$$

4 Numerical results

The accuracy of the TCP throughput method presented in this paper has been evaluated with a numerical example. A simulation scenario has been created based on a topology consisting of one RNC and one Node-B. It is assumed that there is only one MAC-d flow and one priority queue per HSDPA user. The scheduler is Proportional Fair Scheduler. The number of HARQ processes is six; the maximum number of MAC-hs retransmissions is three, whereas the maximum number of RLC retransmissions is six. The number of HS-DSCH codes per cell is five; code multiplexing is not implemented. HSDPA users are connected to the Node-B via HS-DSCH in downlink and via DCH in uplink.

Profile	Ped-A
Speed	3 km/h
Distance	400 m
Trace length	900 s

Table 2: The parameters of the air interface profile

Each HSDPA UE is of category 5/6. The Iub interface and the user plane of the Radio Layer protocols (MAC-d, MAC-hs, RLC, PDCP) are implemented in detail. The transport network of the Iub consists of one CBR VCC. HSDPA users are originating file (ftp) downloads from servers located on the Internet. The transport protocol was TCP Reno; the maximum advertised window size was 48 kbytes; the maximum TCP/IP packet size was set to 1500 bytes. The HSDPA UE reports the observed channel quality (CQI) to the Node-B. Based on this, the amount of data to be sent to the UE is defined. The radio channel condition is simulated separately for each UE. Negative – when the Silence to Noise Ratio (SNR) is below the required threshold – or positive acknowledgement is generated upon reception of a MAC-hs frame. The CQI estimation error is modelled with a constant delay of 6 ms. Users are modelled with ITU-T Pedestrian A model, velocity 3 km/h, assuming that chase-combining is implemented in the UEs. The distance of the users from the Node-B was set to 400 m. The SNR is calculated considering the followings: distance loss according to Okumara-Hata model for urban cell with base station antenna height of 30 m, mobile antenna height of 1.5 m and carrier frequency of 1950 Mhz [8]; multi-path (fast) fading; Rake receiver assuming that channel estimation is ideal and the power levels of all paths are known; shadow (slow) fading (log-normal distribution correlated in time [5]); constant Node-B antenna gain constant (17 dBi); inter-cell interference (−70 dBm) and intra-cell interference (30 dBm).

In the analytical model the air interface trace file has been generated with the MATLAB scripts of the Eurane project (see [6]) with parameters defined by Table 2.

Next, the distribution of the number of RLC PDUs that can be transmitted by the Node-B (denoted by \hat{S} in the paper) is extracted and the analysis method is executed at several link capacity settings (Figure 3).

The figure confirms that the error of the approximation is below 10%. The most important application of analytical throughput computation methods like the one presented in this paper is the transport link dimensioning. During the link dimensioning, the mean throughput (Figure 3) is calculated and the optimal transport link capacity is selected that guarantees the required level of service. The optimal link capacity is around the knee point i.e. where the TCP throughput curve in function of the transport capacity becomes horizontal. Above this point the increase of link capacity does not introduce an increase in the TCP throughput, while below this point the air interface can be underutilized. The optimal link capacity obtained from the analysis and simulation are close to each other, thus our method can be used for transport link dimensioning with a lower computational effort compared to simulations.

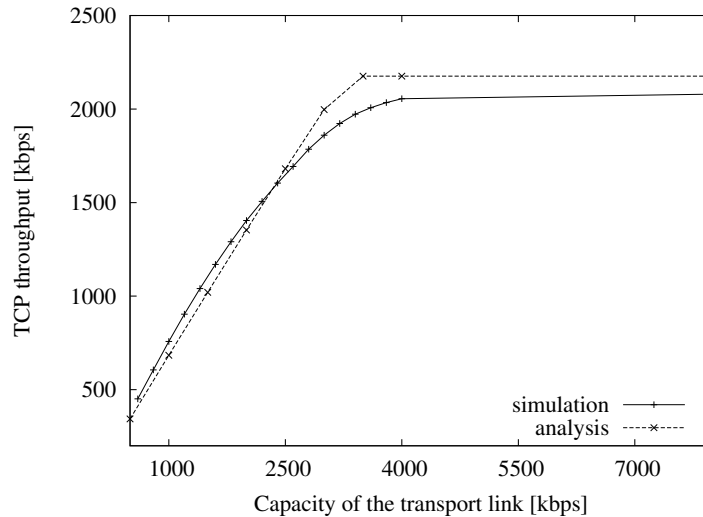


Figure 3: Comparison of the analysis and simulation results

5 Conclusion

In this paper we described an approximation model of the TCP throughput over HSDPA. We identified the relevant congestion points in the system that are having dominant impact on the TCP throughput and developed Markov models to calculate the performance measures. An iterative solution method is provided to solve the queueing network model of the system. The model have been evaluated with a numerical example to evaluate their accuracy and to show that it can be used for the transport link capacity dimensioning of the mobile backhaul.

References

- [1] Mohamad Assaad, Badii Jouaber, and Djamal Zeghlache. Effect of TCP on UMTS-HSDPA System Performance and Capacity. In *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, volume 6, pages 4104–4108, Dallas, TX, USA, November 2004.
- [2] Mohamad Assaad and Djamal Zeghlache. Cross-layer Design in HSDPA System to Reduce TCP Effect. *IEEE Journal on Selected Areas in Communications*, 24(3):614–625, March 2006.
- [3] Mohamad Assaad and Djamal Zeghlache. *TCP Performance Over UMTS-HSDPA Systems*. Auerbach Publications, Boston, MA, USA, 2006.
- [4] Gunter Bolch, Hermann de Meer, Stefan Greiner, and Kishor S. Trivedi. *Queueing Networks and Markov Chains : Modeling and Performance Evaluation With Computer Science Applications*. Wiley-Interscience, August 1998.

- [5] F. Brouwer, I. de Bruin, J. C. Silva, N. Suoto, F. Cercas, and A. Correia. Usage of Link-level Performance Indicators for HSDPA Network-Level Simulations in E-UMTS. In *Proceedings of IEEE ISSSTA '04*.
- [6] Eurane. The eurane project, 2004. <http://www.ti-wmc.nl/eurane/>.
- [7] H. Holma and A. Toskala. *HSDPA/HSUPA for UMTS*. John Wiley & Sons, 2006.
- [8] Harri Holma and Antti Toskala, editors. *WCDMA for UMTS*. John Wiley & Sons, Inc., New York, NY, USA, 2002.
- [9] G. Horváth and Cs. Vulkán. Throughput Analysis of the Proportional Fair Scheduler in HSDPA. In Jan Sykora, editor, *Proceedings European Wireless 2008 (EW2008)*, 2008.
- [10] P.J. Legg. Optimised Iub Flow Control for UMTS HSDPA. *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, 4:2389–2393 Vol. 4, 30 May-1 June 2005.
- [11] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: a Simple Model and its Empirical Validation. In *SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, New York, NY, USA, September 1998. ACM Press.

Rare Event Simulation of Stochastic Activity Networks Using Partition of the Region Technique

Amir Jalaly Bidgoly* Mohammad Abdollahi Azgomi†

Abstract

Modelling and evaluation is an important step of the development of complex systems. In most cases, discrete-event simulation is the only technique that can be used for this purpose. In highly dependable systems, simulation of rare events, corresponding to the failures of system components, will be quite time consuming. A number of techniques are proposed to speed-up simulation of rare events. This paper presents a novel approach for fast simulation of rare events modelled as *stochastic activity networks* (SANs). SANs are a stochastic extension of Petri nets. The solution is based on *partition of the region* (POR) technique and *stratified sampling* method. We have evaluated the proposed method using several examples. Results show that the simulation time is considerably decreased, comparing to the naïve simulation (up to 100 times) and to the *importance sampling* technique. Moreover, the relative error of the simulation results is declined considerably. The proposed method is not dedicated to SANs and can be used for rare event simulation of the other extensions of Petri nets.

1 Introduction

Modelling and evaluation of real-world systems, needs analysing a large and complex model. Analytic techniques can be used to solve a wide range of such models, which are based on state space methods. State space generation is not possible for most models due to the state space explosion problem. In such cases, discrete-event simulation is the only possible technique.

In highly dependable systems that contain rare events, the cost of simulation will increase considerably. The main problem in simulation of highly dependable systems is the small probabilities associated to some important events. These small probabilities make the simulation to be run too long, because of the small times spent in these important rare events.

Rare event simulation is a key tool in some several areas such as reliability evaluation, telecommunications networks, switching systems and similar areas [3, 15, 16]. One famous approach for solving this problem is a technique called *importance sampling* (IS) [4, 12, 16] that is a Monte Carlo simulation variance reduction technique. In typical rare event setting, Monte Carlo method is not viable unless an *acceleration technique* is used to help rare events to occur more frequently. Another

* Performance and Dependability Engineering Laboratory, Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, amir.jalaly@gmail.com

† Performance and Dependability Engineering Laboratory, Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, azgomi@iust.ac.ir

acceleration frequency used for this purpose is *splitting* [11, 18, 37]. IS increases the probability of rare events by changing the probability laws to help the simulation to run faster. Also, it forces the simulation model to focus on rare events. Then, it multiplies the estimator in *likelihood ratio* to correct the result and get an unbiased estimator. The likelihood ratio is roughly the ratio of the original measure and the new measure associated with the generated path. In this way, the cost of evaluation will decrease and will become more acceptable. The main problem in general IS is to find a good probability law. This problem is defined in [15, 31, 4] as a good governor for the model. Unfortunately, a large part of rare-event simulation is focused on *static importance sampling* techniques. This means that a fixed change of measure is used throughout the simulation. While some literature is focused on to the *adaptive importance sampling* technique that changes the measure based on sample simulation path [2].

IS provides several methods that are different in efficiency and are suitable for various problems. Lewis and Bohm has tested IS on Markovian unreliability models and developed *failure biasing* and *forced transition* [17]. Later, Goyal et al. extended the above method in SAVE language [13]. SAVE is basically a generalised machine repairman model. Today several modelling tools provide IS [3031]. One of these tools is the *UltraSAN* [24, 25], which was used in 1990s for modeling and evaluation with *stochastic activity networks* (SANs) [20, 29]. SANs are a stochastic extension of Petri nets. These models have widely been used for performance and dependability evaluation in a wide range of systems.

L'Ecuyer and Tuffin has tried to improve IS by using *bounded relative error* (BRE) and *logarithmic efficiency* (LE) [18, 33, 34].

Splitting technique is also proposed as another approach for rare event simulation. *RESTART*[‡] [36, 35, 37] is a technique based on splitting. This method does not require changing the probability laws for acceleration, but an artificial drift toward the rare event is created by terminating with some probability

trajectories that seems to go away from it and by splitting (cloning) those that are going in the right direction [19, 11, 9, 5]. The main idea in RESTART is repetition of important parts of the system (usually rare event) and getting a higher efficiency on these parts. This method is implemented in ASTRO [36]

This paper presents a novel idea for fast simulation of rare events in SAN models. The solution is based on *partition of the region* (POR) technique [27] as an extension of the *stratified sampling*. A variant of stratified sampling called *transition splitting* has been published in [10]. This technique is extremely efficient on models like an M/M/1 queuing model. The reason is that it uses all available exact knowledge and leaves very little to be simulated. It is very hard to find an appropriate transition splitting and to calculate the probability of each stratum.

Our new solution generally can be used for SAN models. The method is tested on four SAN models and the results and simulation time has been compared to the traditional discrete-event simulation. The results show that simulation is run in the shorter time length while the relative error of the results shows up to 100 times improvements.

The remainder of this paper is organized as follows: In section 2, some background theories are reviewed. In section 3, a new solution for simulation of SAN models

[‡] *REpeated Simulation Trials After Reaching Thresholds*

based on POR technique is presented. In section 4, the results of four example models and analysis of the effect of the proposed method is presented. Finally, some concluding remarks are mentioned in section 5.

2 Background

In this section, we briefly review the background theories and techniques, which are used in the remainder of this paper.

2.1 Stratified Sampling

Let us consider the problem of estimating the below integral:

$$I = \int g(x)dx, \quad x \in D \subset R^n \quad (1)$$

Let suppose that $g \in L^2(x)$ so $\int g(x)^2 dx$ exists and therefore that I exists. For stratified sampling must break the region D into m disjoint subregions $D_i, i = 1, 2, \dots, m$ that is:

$$D = \bigcup_{i=1}^m D_i, D_k \cap D_j = \emptyset, k \neq j.$$

Then let define:

$$I_i = \int_{D_i} g(x)f_x(x)dx, \quad (2)$$

I_i can be estimated separately.

The idea of this technology is similar to the idea of IS: simulation also take more observation in the parts of the region D that are more "important", but the effect of reducing the variance is achieved by concentrating more sample in more important subsets D_i , rather than by choosing the optimal probability density function (PDF).

2.2 Partition of the Region

This technique is similar to stratified sampling and may be able to present an extension for that [27]. In this technique we break the region D into two parts $D=D_1 \cup D_2$, representing the integral I defined in (1) as:

$$I = \int_D g(x)dx = \int_{D_1} g(x)dx + \int_{D_2} g(x)dx. \quad (3)$$

Let us assume that the integral:

$$I_1 = \int_{D_1} g(x)dx \quad (4)$$

can be calculated analytically. Let us define a truncated PDF as:

$$h(x) = \begin{cases} \frac{f_x(x)}{1-P}, & \text{if } x \in D_2 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where:

$$P = \int_{D_1} f_x(x)dx \quad (6)$$

By applying above formulas $h(x)$ became an acceptable PDF. Formula (3) can be written as:

$$\begin{aligned}
I &= I_1 + \int_{D_2} g(x) dx \\
&= I_1 + \int_{D_2} \frac{g(x)}{h(x)} h(x) dx \\
&= I_1 + E \left[\frac{g(X)}{h(X)} \right] \\
&= I_1 + (1-P) E \left[\frac{g(X)}{f_X(X)} \right]
\end{aligned} \tag{7}$$

An unbiased estimated of I is:

$$Y = I_1 + (1-P) \frac{g(X)}{f_X(X)} \tag{8}$$

And the integral I can be estimated by:

$$\theta = I_1 + (1-P) \frac{1}{N} \sum_{i=1}^N \frac{g(X_i)}{f_X(X_i)} \tag{9}$$

This estimator can be used for simulation need some accelerator such as rare event simulation. The variance decrease depended on how break the region.

In section 3, we will try to introduce a method that breaks simulation area in a SAN model in such a way that all unimportant and important events divide into two separate parts. *Important events* are those events that take a role in computation of rare events. Similarly, *unimportant events* are those events that have not any role in computation of rare events.

2.3 Stochastic Activity Networks

Stochastic activity networks (SANs) [20, 29] are a general and stochastic extension of the Petri nets. SANs are powerful and flexible models for concurrent and distributed systems. These models are supported by several powerful modelling tools, such as *UltraSAN* or *Möbius* [8].

Elements of SANs are *places*, *gates* and *activities* as in Petri nets [20]. Places are same as Petri nets. Gates are used to connect places and activities and have two types: *Input* and *output*. Input gates connect one or more places to a single activity and have a *predicate* and a *function*. Output gates connect an activity to one or more places. These gates have only a function. The other element of SAN is activity. There are two types of activities: *timed* and *instantaneous*. Timed activities against instantaneous activities have a delay to *complete*. Delay time represented by a distribution function called *activity time distribution*. $F(\cdot, \mu; a)$ denotes the distribution function for activity a in marking μ . Activities also have *cases* in their outputs to show uncertainly about action taken at completing. $C(\cdot, \mu; a)$ used for showing *case distribution function* of activity a in marking μ .

An activity called *enabled* when all gates and places connected directly to activity hold *true*. The predicate of gates must return true and places must has at least one token. When an activity is enabled, it waits for completing. Time for delay taken from time activity distribution called *activity time*. After completion, first the input functions executed and then the output function. An enabled activity does not require completion. During activity time, SAN can move to a marking that activity be no more enabled and *aborted*.

3 Fast Simulation of SANs Using POR Technique

Consider a SAN model with possible path set Ω and possible events set Π . Our interest is in estimating a probability $P(\varepsilon)$ of a rare event $\varepsilon \in \Pi$. Let $I(\varepsilon)$ be an *indicator function*, ε , which is defined as follows:

$$\varepsilon = \begin{cases} 1 & \text{if the events belong to } \varepsilon \text{ are simulated in the model} \\ 0 & \text{otherwise} \end{cases}$$

In practice, a *reward variable* can be defined for the indicator function. Let γ denote the probability $P(\varepsilon)$. This may be estimated by Monte Carlo i.e. generating n independent sample of I ($I_1(\varepsilon), I_2(\varepsilon), \dots, I_n(\varepsilon)$) and taking the average $\frac{1}{n} \sum_{i=1}^n I_i(\varepsilon)$ as

estimator of $P(\varepsilon)$ called γ_n . In general Monte Carlo method, when $P(\varepsilon) \rightarrow 0$ for reaching $\gamma_n \rightarrow \gamma$ almost sure as $n \rightarrow \infty$. This cause to relative error (or relative variance) be constant or at least be bounded [15, 18].

First try to solve this problem need to decrease variance (related error or related variance). For getting best result variance must be zero [16]. This needs $I(\varepsilon)$ be equal to one in every simulation run. However, this is not possible to be implemented. In practice, this guides us to run the model in a way that the probability of executing rare events increases. One approach is using POR such that only the important part of model is simulated, which means rare events in a rare event simulation.

In this case first must find a way that break the path set (here a SAN model). Partitioning must keep all rare events in one part that is the goal of simulation. If we can break model such that the second part has no rare event ($I(\varepsilon)$ always be zero) so no need to simulate this part. This help run time and efficiency get dramatic results.

Let us redefine Ω as follows:

$$\begin{aligned} \Omega = & \{ \text{all path in model} | \\ & \text{start state of path} \\ & = \text{final state of path} = x_0 \} \end{aligned} \quad (10)$$

where, x_0 is the initial state of the model.

In the next step, we need to break Ω set into two separate sets, as below:

$$\begin{aligned} \Omega_1 &= \{ \chi \in \Omega | I(\varepsilon) = \text{constant in } \chi \} \\ \Omega_2 &= \{ \chi \in \Omega | \chi \notin \Omega_1 \} \end{aligned} \quad (11)$$

which are our new state space sets.

Note that only the second set is important for us, which contains rare events. In practice, the place that is the beginning of Ω_1 and Ω_2 paths is chosen as the start state, i.e. the last common place that belongs to the intersection of the two sets.

In the next step, the SAN model will be partitioned into two separate parts. For this purpose, the first part must be divided into events set Π . Let us define two new sets as follows:

$$\begin{aligned} \Pi_1 &= \{ e \in \Pi | \nexists \chi \in \Omega_2 | e \text{ is in } \chi \} \\ \Pi_2 &= \{ e \in \Pi | \exists \chi \in \Omega_2 | e \text{ is in } \chi \} = \\ & \Pi - \Pi_1 \end{aligned} \quad (12)$$

In simple words if event e is being ran in simulation of a path x , e is a member of Π_2 if and only if there is some x that belong to Ω_2 . Otherwise e is a member of Π_1 . If find some $x_1 \in \Omega_1$ and $x_2 \in \Omega_2$ e will be in Π_2 . So, common events between two path sets will be in Π_2 respectively.

Final step is computing P as defined in (6). Let us define a new indicator function:

$$I(\varepsilon') = \begin{cases} 1 & \varepsilon' \in \Pi_1 \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

A simple simulation can evaluate $P(\varepsilon')$. ε' has no rare event and a general Monte Carlo method take result with good confidence level.

$$\hat{\gamma}_n(P') = \frac{1}{n} \sum I(\varepsilon'). \quad (14)$$

In practice, $I(\varepsilon')$ implement via a reward variable and it is large enough along simulation to compute in normal executing.

Then it is easy to break SAN model. Just remove all events belong to Π_1 . All remained events are rare or need to run rare events. Definitely it is reduce variance to dramatic small value. In continue just use the below estimator:

$$\hat{\gamma}_n(P) = \frac{1}{n} \sum I(\varepsilon) \cdot (1 - P') \quad (15)$$

that take random variable only from important space of model Ω_2 . It is clear that

$$\int_{\Omega_1} I(\varepsilon) d\varepsilon = 0 \quad (16)$$

This help remaining part simulate correctly respect to rare event evaluation.

Example. A SAN model is shown in Figure 1, which models a rare event with a low distribution of 0.0001. In this model Ω simply can be defined as

$$D = \{(rare - event, act1, act2), (act3, act4, act5)\}; \quad (17)$$

place1 has one token in the initial state and all other places are empty. So, initial state is $[1, 0, 0, 0, 0]$. Now let us consider indicator function point to token in *place3* that is a result of completing rare event activity. Ω_1 and Ω_2 respectively define as

$$\begin{aligned} \Omega_1 &= \{(act3, act4, act5)\} \\ \Omega_2 &= \{(rare - event, act1, act2)\} \end{aligned} \quad (18)$$

So, Π_1 and Π_2 will be

$$\begin{aligned} \Pi_1 &= \{act3, act4, act5\} \\ \Pi_2 &= \{rare - event, act1, act2\} \end{aligned} \quad (19)$$

It is clear that $I(\varepsilon')$ points to completing of *act3*, *act4*, *act5*. Also clear that P' simply can be compute.

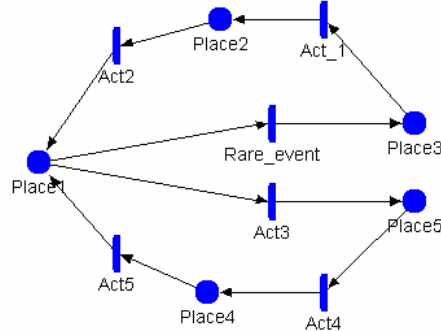


Figure 1. A SAN model with a rare event

Table 1. Activities of SAN model in Figure 1

Timed Activity	Distribution Parameters
<i>Act3</i>	Exponential 100
<i>Act4</i>	Exponential 10
<i>Act5</i>	Exponential 10
<i>Act1</i>	Exponential 10
<i>Rare_Event</i>	Exponential 0.0001

Just as last step for using POR we define probability P' as required in (15). Let us define a new reward variable R' as

$$R = \begin{cases} \sum_{p \in N} mark(p) & \text{reward-function} \\ 0 & \text{impluse-function} \end{cases} \quad (20)$$

That p is a place in set $\{place4, place5\}$ (an unimportant part) and $mark(p)$ return number of token in p . For computing R' SAN model must simulate in normal mode. In this step fast simulation does not require, because R' does not contain any rare event. Result of R' is the value of probability P' defined in (15). An Impulse reward can be used for this purpose respectively. However we don't define that for simpler implementation.

As POR shows model breaks into two parts Ω_1 and Ω_2 . Ω_2 is the important part and must observe more than Ω_1 . Ω_1 can simulate normally But about Ω_2 that contain goal of simulation probability P' help us to estimate result in correct mode therefore we have a new SAN model that only contain set Ω_2 and Π_2 . New SAN model of Figure 1 is shown in Figure 2.

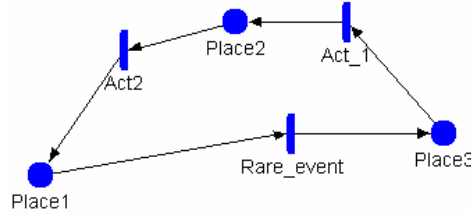


Figure 2. New SAN model of Figure 1 for using POR

4 Applications and Results

In this section we present the results of using our proposed method on four sample SAN models. To evaluate the proposed method, we have used the Möbius modelling tool. First, a simple SAN model with only one rare event is simulated. Then, another model with some other normal events is tested. Finally, a third model with two rare events and some interesting properties is tested. And finally, we have chosen a sample to compare the proposed method with the IS technique.

4.1 Example 1: A Simple Model

As the first example a simple SAN model is tested. This model contains only a rare event with a normal event that will race together. The SAN Model is shown in Figure 3 and its properties in Table 2 and Table 3.

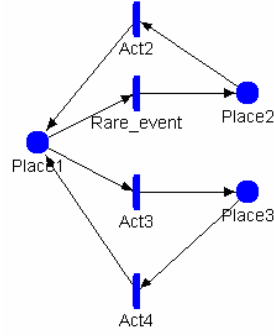


Figure 3. A simple SAN model with one rare event

Table 2. Initial markings of SAN model of Figure 3

Place Names	Initial Markings
Place1	1
Place2	0
Place3	0

In first step Ω_1 and Ω_2 must define:

$$\begin{aligned}\Omega_1 &= \{(act3, act4)\} \\ \Omega_2 &= \{(rare-event, act2)\}\end{aligned}\quad (21)$$

And

$$\begin{aligned}\Pi_1 &= \{act3, act4\} \\ \Pi_2 &= \{rare-event, act2\}\end{aligned}\quad (22)$$

Table 3. Activities of SAN model of Figure 3

Timed Activity	Distribution Parameters
Act2	Exponential 10
Act3	Exponential 100
Act4	Exponential 10
Rare_Event	Exponential 0.00001

In next step we define a reward variable as (23) for computing probability P' . This reward variable is simply defined on Ω_1 set.

$$R = \begin{cases} \text{mark}(\text{place3}) & \text{reward-function} \\ 0 & \text{impluse-function} \end{cases} \quad (23)$$

We evaluate this reward variable by Möbius modelling tool and get the result as 9.090908e-001. Now we test the original model and our new model, which contains only *Place1* and *Place2* for the result of the reward variable defined in (24) (i.e. removing *Act3* activity).

$$R = \begin{cases} \text{mark}(\text{place2}) & \text{reward-function} \\ 0 & \text{impluse-function} \end{cases} \quad (24)$$

Simply this new model is contains only Π_2 set Simulation results of this model for evaluation of rare event are shown in table 5. Simulator uses $(1-P')$ to biased the estimator. Remember that goal is rare event or in the other word token in *place2*.

Simulator run both traditional and POR methods. The time is in terms of seconds. The results represent huge improvements in simulation time and precision. The results are compared to the outputs of the *steady state solver* of Möbius modelling tool. This helps us to get a real relative error that shown in last column.

Table 4. Results of simulation of SAN model represnted of Figure 3

Method	1-P	Time(s)	Results	Replications	Confidence Interval	Error
Naive Simulation	-	7267.404	9.6558236535E-08	240636000	1.0887829753E-08	6.21%
POR	0.09090809	186.843	9.6199450981E-08	69213000	9.6065787077E-09	5.81%

4.2 Example 2: A More Complex Example

In this example, the model has some other normal events running with rare event, so model is more than just two simple parts. In this model still there is no event commonly in Ω_1 and Ω_2 . The model presented in Figure 4 obviously has two separate parts:

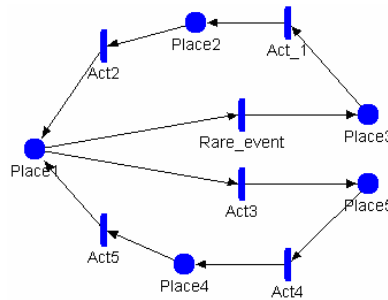


Figure 4. A SAN model with a rare event

Table 5. Results of simulation of SAN model represnted of Figure 4

Method	1-P	Time(s)	Results	Replications	Confidence Interval	Error
Naive simulation	-	2012.609	5.0371920274E-07	67975000	5.0284734014E-08	5.78%
POR	0.04761813	42.922	4.6574525135E-07	12197000	4.6561591212E-08	2.19%

Table 6. Activities of SAN model of Figure 4

Timed Activity	Distribution Parameters
Act3	Exponential 100
Act4	Exponential 10
Act5	Exponential 10
Act1	Exponential 10
Rare event	Exponential 0.0001

In the model presented in Figure 4, the initial markings of all places except *Place1* are zero. *Place1* has one token in its initial markings. The sets $(\Omega_1, \Omega_2, \Pi_1, \Pi_2)$ of this model have shown in (18) and (19) so just see the result of simulation in table 6. Results in this test are also compared with outputs of steady state solver of Möbius software.

4.3 Example 3: Two Rare Events

As third example, we choose a model with two rare events. This model has a special property that make it distinguished from previous samples. In the presented model, Ω_1 and Ω_2 sets have some common events in their members. We interested in this model from this view that only just one activity is removed for fast simulation by our method however all places still cooperate in simulation. Notice that in previous sample some places goes away from simulation. The model is shown in Figure 5. The indicator function points to fail events so; the goal is computing the following reward variable:

$$R = \begin{cases} \text{mark}(\text{fail1}) + & \text{reward} - \text{function} \\ \text{mark}(\text{fail2}) & \\ 0 & \text{impluse} - \text{function} \end{cases} \quad (25)$$

Sets Ω_1, Ω_2 are as below

$$\begin{aligned} \Omega_1 &= \{(\text{job_request}, \text{job_done})\} \\ \Omega_2 &= \{(\text{job_request}, \text{fail1}, \text{repair1}), \\ &\quad (\text{job_request}, \text{fail2}, \text{repair2})\} \end{aligned} \quad (26)$$

In the previous examples, the start state of paths is same as the start state defined for model, but in this model to simplify the paths we have changed the sets as bellow:

$$\begin{aligned} \Omega_1 &= \{(\text{job_done}, \text{job_request})\} \\ \Omega_2 &= \{(\text{fail1}, \text{repair1}, \text{job_request}), \\ &\quad (\text{fail2}, \text{repair2}, \text{job_request})\} \end{aligned} \quad (27)$$

It means that the start state is $(\text{job_doing}, 1)$ marking (the last common state between two sets). In continue we define

$$\begin{aligned} \Pi_1 &= \{\text{job_done}\} \\ \Pi_2 &= \{\text{job_request}, \text{fail1}, \text{fail2}, \text{repair1}, \text{repair2}\} \end{aligned} \quad (28)$$

As seen in (28) job_request is common between two path sets however at last it moved to Π_2 set. Indicator of ε' points to completing of activity job_done and job_request in sequence. Simply because completing sequence $\text{fail}, \text{repair}, \text{job_request}$ is a rare event, P can computing by supposing the all tokens in idle place is moved in by completing job_done activity. A reward variable can be defined for this purpose. Definition of this variable is as follows:

$$R = \begin{cases} \text{mark}(\text{idle}) & \text{reward} - \text{function} \\ 0 & \text{impluse} - \text{function} \end{cases} \quad (29)$$

Probability of P' is the value of evaluating of the variable in (29). However this reward variable is not actually as same as analytic definition of indicator of ε' but in practice they are such close that can be assume one. Now model can be simulated easily by removing job_done activity and using $1-P'$ probability for biasing the estimator. Result of this simulation is shown in table (7). This model is compared with results of steady-state solver of Möbius modelling tool.

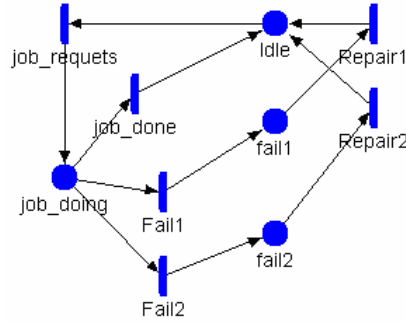


Figure 5. A SAN model with two rare events

Table 7. Initial markings of SAN model of Figure 5

Place Names	Initial Markings
<i>idle</i>	1
<i>Fail1</i>	0
<i>Fail2</i>	0
<i>Job doing</i>	0

Table 8. Activity of SAN model of Figure 5

Timed Activity	Distribution Parameters
<i>Job request</i>	Exponential 100
<i>Job done</i>	Exponential 1000
<i>Fail1</i>	Exponential 0.0001
<i>Fail2</i>	Exponential 0.00001
<i>Repair1</i>	Exponential 1
<i>Repair2</i>	Exponential 10

Table 9. Results of simulation of SAN model represented of Figure 5

Method	I-P	Time(s)	Results	Replications	Confidence Interval	Error
Naive simulation	-	2718.750	8.7378789281E-06	13210000	8.7044577598E-07	4.83%
POR	0.09092855	788.547	9.1541250070E-06	100000000	2.0061251233E-07	0.29%

4.4 Example 4: POR vs. IS

For last example of presented method, we test that on a model studied by Obal II and Sanders [26] for IS technique presenting in *UltraSAN*. This model can be seen in Figure 6. Obal II study the unreliability of this model over an interval of time. This can be computed through an *instant of time reward variable* when the system failure marking is an absorbing marking by examining the instantaneous rate reward at the end of interval [26]. Model is a *machine-repairman system* that uses a delayed group repair policy. There are two types of components in the system, with different failure rates. The places labeled type 1 and type 2 model the two types of components. The marking of each place represents the number of working components of that type. In this case, there are two *type-one* components, and four *type-two* components. Timed activities *fail_1* and *fail_2* model the time between failures for each component. As shown in Table 11, the failure times are exponentially distributed with marking

dependent rate parameters. A component of type-one fails with rate 0.005, while components of type-two fail at twice that rate. The use of marking dependent rate parameters allows us to avoid including an activity for each component's failure time distribution, resulting in a more compact representation.

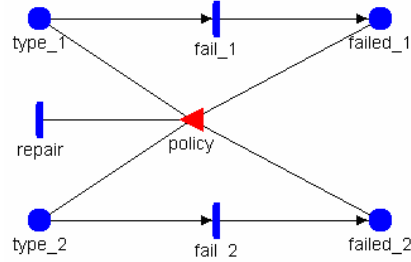


Figure 6. The *machine-repairman* model presented by Obal II

Table 10. Initial markings of SAN model of Figure 6

Place Names	Initial Markings
<i>type_1</i>	2
<i>type_2</i>	4
<i>failed_1</i>	0
<i>failed_2</i>	0

Table 11. Activity of SAN model of Figure 6

Timed Activity	Distribution Parameters
<i>fail_1</i>	Exponential $0.005 * \text{Mark}(\text{type_1})$
<i>fail_2</i>	Exponential $0.01 * \text{Mark}(\text{type_2})$
<i>repair</i>	Exponential 1

Table 12. Activity of SAN model of Figure 6

Gate	Enable Predicate	Function
<i>policy</i>	$(\text{type_1} \rightarrow \text{Mark}() > 0 \parallel \text{type_2} \rightarrow \text{Mark}() > 0) \ \&\& \ (\text{failed_1} \rightarrow \text{Mark}() == 2 \parallel \text{failed_2} \rightarrow \text{Mark}() > 1)$	<pre> if (failed_1->Mark() == 2) { failed_1->Mark()=0; type_1->Mark() = 2; } else { failed_2->Mark() = 0; type_2->Mark() = 4; } </pre>

The markings of places *failed_1* and *failed_2* represent the number of components of each type that have failed. There is one repairman in the system. The repair policy is to wait until at least two components of the same type have failed, and then begin to repair the whole group. Type-one component repair is given preemptive priority over repair of type-two components. When the repair is completed, all components of that type are as good as new. This repair policy is implemented in the input gate *policy*. The properties of *policy* are shown in table 12. *type_1*->Mark() points to marking of *type_1* and so on. If all components of both types fail, the system fails, and all repair activity halts; the failed state is an absorbing state.

For evaluation of unreliability of model a reward variable is used. A reward structure that identify the failed state is

$$R = \begin{cases} 1 & \text{if } (type_1, 0) \text{ and } (type_2, 0) \\ 0 & \text{otherwise} \\ 0 & \text{impluse-function} \end{cases} \quad \text{reward-function} \quad (30)$$

Goal is computing the unreliability in interval $[0, 100]$. This can be done by using reward variable as instant of time reward variable.

For using this model in POR method, we must break it into two parts. For this purpose we define two type of repair: *repair_type_1* and *repair_type_2*. Each repair type note to repairing of one type e.g. *repair_type_1* is repair activity on type one components. Also as described before the start state for defining cycle is the last marking common between two parts. This state simply is $\{(failed_1, 2), (failed_2, 4)\}$ when system fails and $\{(failed_1, 2)\}$ or $\{(failed_2, 2)\}$ when *repair* enabled. So sets Ω_1 , Ω_2 are as below

$$\begin{aligned} \Omega_1 &= \{(repair_type_1, fail1, fail1), (repair_type_2, fail2, fail2), \dots\} \\ \Omega_2 &= \{(systemfail)\} \end{aligned} \quad (31)$$

Ω_1 members are those who run *repair* activity however Ω_2 has only one member that is fail state. If we continue this partitioning, for computing P we must evaluate following probability when a component fail.

$$1 - P = \frac{P(\Omega_2)}{P(\Omega_1) + P(\Omega_2)} = \frac{P(systemfail)}{P(systemfail) + P(repair)} \quad (32)$$

This probability is a rare event and is same as finding unreliability. So in this model we use some other Ω sets. These new sets however does not complete push rare events on one part and other events in other part, help simulation in a way to reduce time and variance. New sets defined as

$$\begin{aligned} \Omega_1 &= \{(repair_type_1, \dots)\} \\ \Omega_2 &= \{(systemfail), (repair_type_2, \dots)\} \end{aligned} \quad (33)$$

This partitioning breaks system into one part that contain repairing of type one components and another part has fail state and repairing of type two components. Now $1-P$ define as

$$1 - P = \frac{P(\Omega_2)}{P(\Omega_1) + P(\Omega_2)} = \frac{P(systemfail) + P(repair_type_2)}{P(systemfail) + P(repair_type_1) + P(repair_type_2)} \quad (34)$$

Probability (32) can evaluate simply. Partitioned model has no repair activity for type one components. This increase the frequently of system fail while $(1-P)$ probability help for biasing the estimator. The result of this simulation is shown in table 14. Results of Obal II simulation also present in table 13.

Table 13. Results of simulation of SAN model represnted of Figure 6

Method	Time(s)	Results	Replications	Error
Naïve simulation	67230	1.52E-06	28375000	20.8%
IS	1385	1.90E-06	338497	1.02%

Table 14. Results of simulation of SAN model represnted of Figure 6

Method	1-P	Time(s)	Results	Replications	Error
Naïve Simulation	-	1290.203	1.7176054559E-06	79180000	10.5%
POR	0.1147283	73.203	1.9193768592E-06	8189000	0.01%

Both simulation (naïve and partitioned) run with 98% confidence level. Result in our simulation show about 17 times improvement in simulation time and about 1050 times in relative error while Obal II's results shows 48 in former and 20 times in later, respectively. Note that in this model, POR is not used completely because of reaching another rare event when computing the probability P . The problem of using the presented method for this model is that common part between Ω_1 , Ω_2 is not so much so computing P is need computing original rare event. However, Simulation results' show good improvements.

5 Conclusions

Partition of the region by breaking the region of the simulation into two parts helps simulator to spend more time on important parts of the model. In this paper, using this technique a new approach for fast simulation of SAN models is introduced. For this purpose, we have defined sets Ω_1 and Ω_2 than breaking a model using Π_1 and Π_2 sets. Then, the probability of being in the important part that is defined by a new indicator function is computed. And finally, the model is simulated without events in set Π_2 .

Since partition of the region uses simulation in a way that only rare events are observed, it improves efficiency of simulation. Also this technique can simply define dynamically so one can enjoy this method automatically on every SAN model.

We have evaluated the proposed method using four examples of SANs. Results show that simulation time is decreased even up to 100 times, while the results of the simulation show high improvements regarding the relative errors.

The method presented in this paper is not dedicated to SANs and can also be used with other stochastic extensions of Petri nets, such as SPNs, GSPNs, etc. It is also possible to use this method with Markov chains. We are currently working to use the proposed method for rare event simulation of SPNs and Markov chains.

References

- 1 Blum, A.M. Heidelberger, P. Lavenberg, S.S. Nakayama, M. and Shahabuddin, P., "System availability estimator (SAVE) language reference and user manual version 4.0," Technical Report RA 219 S. IBM Thomas J. Watson Research Center, June 1993.
- 2 Borkar, V. S., S. Juneja, A. A. Kherani. "Performance analysis conditioned on rare event: An adaptive simulation scheme." *Communication in Information*, 3, 4, 259-278, 2004.
- 3 Bulhuis, P. G., Chandler, D., Dellago, C., and Geissler P. L., "Transition path Sampling: Throwing robe over rough mountain passes, in the dark," *Annual Review on physical Chemistry* 53, 2002, pp. 291-318.
- 4 Bucklew, J. A., *Introduction to Rare Event Simulation*, Springer-Verlag, New York, 2004.
- 5 Ce'rou, F., F. LeGland, P. Del Moral, and P. Lezaud, "Limit theorems for the multilevel splitting algorithm in the simulation of rare events", In *Proc. of the 2005 Winter Simulation Conference*, ed. F.B.A.M.E. Kuhl, N.M. Steiger and J.A. Joines, 2005, pp. 682–691.
- 6 Dellago, C. Bolhuis, P. G. Csajka, F. S. and Chandler, D., "Transition Path Sampling and the Calculation of Rate Constants," *J. Chem. Phys.* 108, 1998.
- 7 Dellago, C. Bolhuis, P. and Geissler. P. L., "Transition Path Sampling," *Adv. Chem. Phys* 123, 2002.

- 8 Deavours, D.D., et al., "The Möbius Framework and Its Implementation," IEEE Trans. on Soft. Eng. 28(10), IEEE CS Press, 2002, pp. 956-969
- 9 Garvels, M. J. J., *The splitting method in rare event simulation*, Ph. D. Thesis, Faculty of Mathematical Science, University of Twente, The Netherlands, 2000.
- 10 Gaviroński. A. A. "Transition splitting for estimation of rare events with application to high speed data networks," In Labatoulle and Roberts, 32, 1994, pp. 767–776.
- 11 Glasserman, P., P. Heidelberger, and P. Shahabuddin, "Asymptotically optimal importance sampling and stratification for pricing path dependent options," Mathematical Finance 9 (2), 1999, pp. 117–152.
- 12 Glynn, P. W. and Iglehart, D. L. 1989. "Importance sampling for stochastic simulations." Management Science 35, 1367–1392.
- 13 Goyal, A. Shahabuddin, P. Heidelberger, P. Nicola, V.F. and Glynn, P. W. "A Unified Framework for Simulating Markovian Models of Highly Dependable Systems," IEEE Transactions on Computers, vol. 41, no. 1, Jan. 1992, pp. 36-51.
- 14 Heegaard, P. E., "Speed-up techniques for simulation," Teletronikk, 1995.
- 15 Heidelberger, P., "Fast Simulation of Rare Events in Queuing and Reliability Models," ACM Transactions on Modelling and Computer Simulation, vol. 5(1), 1995, pp. 43-85.
- 16 Juneja, S. and Shahabuddin, P., "Rare event simulation techniques: An introduction and recent advances." In Simulation S. G. Henderson and B. L. Nelson, Eds. Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, The Netherlands, 2006, pp. 291-350
- 17 Lewis, E. E. and Bohm, F., "Monte Carlo of Markov Unreliability Models," Nuclear Engineering and Design 77, pp. 49-62.
- 18 L'Ecuyer, P., Blanchet, J., Tuffin, B., and Glynn, P. W., "Asymptotic Robustness of Estimators in Rare event Simulation", ACM Transactions on Modelling and Computer Simulation, 2007.
- 19 L'Ecuyer, P., Demers, V., and Tuffin, B. "Splitting for Rare event Simulation," In Proc. of the 2006 Winter Simulation Conference, IEEE Press, 2006, 137-148.
- 20 Meyer, J.F. Movaghar, A. and Sanders, W.H., "Stochastic Activity Network: Structure, Behavior and Application," In Proc. of Int'l Workshop on Timed Petri Nets, Torino, Italy, July 1985, pp. 106-115.
- 21 Nicola, V.F. Nakayama, M. K. Heidelberger, P. and Goyal, A., "Fast Simulation of Dependability Models with General Failure, Repair and Maintenance Process," In Proc. of 20th Annual International Symposium on Fault-Tolerance Computing, Newcastle upon Tyne, United Kingdom 1990, pp. 491-498.
- 22 Nicola, V.F. Shahabuddin, P. and Nakayama, M., "Techniques for Fast Simulation of Models of Highly Dependable Systems," In Proc. of the IEEE Transition on Reliability, Vol. 50, No 3, 2001.
- 23 Nicola, V.F. Shahabuddin, P. Heidelberger, P. and P.W. Glynn, "Fast simulation of Steady-State Availability in Non-Markovian Systems," 1993.
- 24 Obal II, W.D. and Sanders, W.H., "An Environment for Importance Sampling Based on Stochastic Activity Networks," In Proceedings of the 13th Symposium on Reliable Distributed Systems, Dana Point, CA, Oct. 1994, pp 64-73.
- 25 Obal II, W. D. and Sanders, W. H., "Importance Sampling Simulation in UltraSAN", From Simulation, Vol. 62, No. 2, Feb. 1994, pp. 98-111.

- 26 Obal II, W.D., "Importance Sampling Based of SAN-Based Reward Models" Master Thesis, The University of Arizona, July 1993.
- 27 Rubinstein, R. Y., "Simulation and Monte Carlo Method," Wiley series in probability and mathematical statistics, John Wiley and sons, New York, United State of America, 1981, pp. 114-158.
- 28 Sanders, W. H., and Meyer, J. F., "A Unified Approach for Specifying Measures of Performance, Dependability, and Performability," in Dependable Computing for Critical Applications, Dependable Computing and Fault-Tolerance Systems, Vol. 4., A. Avizienis and J. C. Laprie, editors, Springer Verlag, Vienna, 1991, pp. 215-238.
- 29 Sanders, W. H., Meyer, J. F., "Stochastic Activity Networks: Formal Definitions and Concepts," Lectures on Formal Methods and Performance Analysis, 2001.
- 30 Shahabuddin, P., "Simulation and Analysis of Highly Dependable Systems," PhD Thesis, Stanford University, 1990.
- 31 Shahabuddin, P., "Importance Sampling for the Simulation of Highly Reliable Markovian Systems," management Science, vol. 41, no. 3, March 1994, pp. 333-352.
- 32 Shreider, Yu. A., "The Monte Carlo Method (the Method of Statistical Trials)," Pergamon, Elmsford, New York, 1966.
- 33 Tuffin, B., "Bounded normal approximation in simulations of highly reliable Markovian systems." Journal of Applied Probability 36, 4, 1999, pp. 974-986.
- 34 Tuffin, B., "On numerical problems in simulations of highly reliable Markovian systems," In Proc. of the 1st International Conference on Quantitative Evaluation of SysTems (QEST). IEEE CS Press, University of Twente, Enschede, The Netherlands, 2004, pp. 156-164.
- 35 Ville`n-Altamirano, M., and Ville`n-Altamirano, J., "RESTART: A Straightforward Method for Fast Simulation of Rare events," In Proceedings of the 1994 Winter Simulation Conference, 1994, pp. 282-289.
- 36 Ville`n-Altamirano, M., and Ville`n-Altamirano, J., "RESTART: A Method for Accelerating Rare Event Simulations," in J. W. Cohen, C.D. Pack, editors, Queuing, Performance and Control in ATM, 13th International Teletraffic Congress, Copenhagen, North-Holland, 1991, pp. 71-76.
- 37 Ville`n-Altamirano M. and Ville`n-Altamirano, J. "On the efficiency of RESTART for multidimensional systems." ACM Transactions on Modeling and Computer Simulation 16, 2006, 3, 251-279.
- 38 Williamson, A., "Discrete Event Simulation in the Möbius Modelling Framework." Master's thesis, University of Illinois at Urbana-Champaign, 1998.

From architecture to SWN models for compositional performance analysis of Component Based Systems: application to CCM based systems

Nabila Salmi*, Patrice Moreaux[†] and Malika Ioualalen[‡]

Abstract

Predicting performance in early stages of software development is an important issue, especially in the case of systems developed as an assembly of components. The analysis of such Component based systems (CBS) may be difficult or impossible to conduct, because of the combinatorial state space explosion. To cope with this phenomenon, the paper proposes an efficient compositional method for modelling and performance analysis of CBS, applied to the CORBA Component Model (CCM). The method starts from the definition of the architecture of the system and its components, and applies a systematic translation into a structured interconnection of formal models (Stochastic Well formed Nets (SWN), a high level model of Stochastic Petri Nets) associated to components and their interactions. We then derive performance indices of the system through an efficient analysis based on the structure previously built.

1 Introduction

The desire to bring better quality and higher efficiency in software design has led to the development of *Component Based Systems (CBS)*. These systems are made of elementary *bricks or components*, assembled together [19]. This approach has attracted both academic and industrial communities in many engineering fields (embedded systems, web-based applications, etc). Several *component models* have been defined for this purpose such as EJB, CCM (CORBA Component Model), .NET, Fractal, PECOS, Koala. For most of these models, an *Architecture Description Language (ADL)* [14] allows to describe an assembly of components. From this description, a set of tools generate the application code and perform some formal verifications such as type compatibilities. Component based technologies provide rapid development and promise significant benefits. However, when assembling components, software designers should have some assurance that the resulting system meets the performance expected by users and avoids contention and bottlenecks. Moreover, for large architectures of CBS, such properties are more difficult to derive. So, it is advisable to have

*LISTIC, Annecy, France, LSI, USTHB, Algiers, Algeria, nabila.salmi@univ-savoie.fr

[†]LISTIC, Université de Savoie, Annecy, France, patrice.moreaux@univ-savoie.fr

[‡]LSI, USTHB, Algiers, Algeria, ioualalen@lsi-usthb.dz

methods and tools that allow qualitative and quantitative analysis of CBS, to support designers in their activities.

To achieve this goal, several work was proposed mainly for qualitative analysis. [1] uses model checking of Labeled Transition Systems (LTS) to prove temporal logic properties of a Fractal CBS. [7] and [17], are respectively based on hierarchical coloured Petri nets (HCPN) and generalized stochastic nets (GSPN). In contrast, performance analysis is usually carried out with measures on existing systems using performance testing [9]. We can however note proposals for predictive performance modelling [4]. This approach, followed by [21, 10], translates architecture designs, mostly given in the UML language into adequate performance models such as Layered Queuing Networks (LQN) [8], Stochastic Petri Nets (SPN) and Stochastic Process Algebras (SPA). However, we claim that UML is not sufficiently expressive to model complex systems, and that QN don't allow synchronization, resource contention and conflicts modelling, which are important characteristics of actual systems.

In this perspective, we propose a structured compositional approach for performance analysis of a CBS, trying as possible to reduce complexity of the analysis. The main idea is to start from the architecture description of the CBS expressed in an ADL, to model components, to derive the CBS global model, and then to apply a structured compositional method to derive performance indices. XXX Components are modelled with Stochastic Well-formed Net (SWN) [3], a special class of Stochastic coloured Petri Nets, that we believe being the most suitable formalism for performance analysis of complex symmetrical systems. SWNs constitute a high level state based model, able to model complex systems with concurrency and conflicts and enabling performance indices evaluation. Moreover, they benefit from a large set of analysis algorithms and tools [13].

Our approach first translates systematically components interfaces and interactions in the SWN context. Two main interaction patterns are defined between components: synchronous request/response interactions provided with the method invocation (such as an RPC or RMI communication) and asynchronous interactions given through notification of events. We then show how to build the global SWN of the CBS. Finally, we apply a structured method, derived from our previous works [5, 6] and adapted to CBS, allowing to compute performance indices in an efficient way. Computations are based on a combined aggregation/tensorial representation of the underlying Markov chain of the global SWN, which reduces the complexity of the analysis (time and memory). In a previous paper [18], we have studied modelling and performance evaluation of Julia implementation of *FRACTAL* based CBS, which provides synchronous interactions between *FRACTAL* components. The present paper extends this work to more general CBS with asynchronous event-based interactions. We have chosen to illustrate our approach with the *Corba Component Model*, *CCM* (CCM-CBS) [15]. CCM is indeed a language-neutral model, using the two classical interaction modes : request/response and event-based.

This paper is organized as follows. Section 2 presents the main features of the CCM model, illustrated by an example of application. We then give details of our method in section 3. We illustrate in section 4, the application of the approach to our example. We conclude and give future work in section 5.

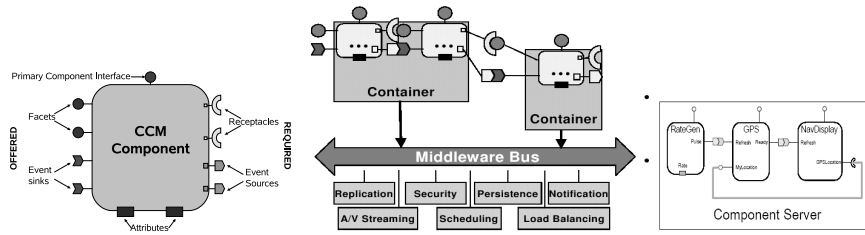


Figure 1: CCM component interfaces (left), CCM based application architecture (middle) and the Avionics control system example (right)

2 The Corba Component Model

The CORBA Component Model (CCM) [15] is a component model independent from operating systems and programming languages, designed to address the limitations with earlier versions of CORBA 2.x middleware.

CCM features A CCM component is an implementation entity, described by a set of *attributes* and a set of communication points termed *ports* or *interfaces* in the sequel. Attributes are named values exposed through accessor and mutator operations, primarily intended to be used for component configuration. Interfaces (figure 1, left) are access points of four types: *facets* and *receptacles* supporting a synchronous-style interaction and *event sources* and *sinks* providing asynchronous event-based communication. *Facets* accept point-to-point invocations from other components; *receptacles* indicate a dependency on point-to-point method interface of another component; *event sources* emit asynchronous messages (events) of a specified type to one or more interested components, and *event sinks* receive from sources events of a specified type. CCM support for events follows the *publish/subscribe event push* model [15], compatible with the CORBA notification service: Sinks, said *subscribers*, register (subscribe) for a class of events published by a source (*publisher*). Intermediate *event channels* are used to broker event messages between publishers and subscribers. A channel manages a specified type of event. It receives an event notification from one publisher and acknowledges it. Then, it sends the notification to all interested subscribers. At reception of an event, a subscriber acknowledges it and runs an *event handler* for processing the event.

A CCM component is located inside a *container* which provides it with the runtime environment and allows it to access a set of system and middleware services such as persistence, transaction, security and event services (figure 1, middle). A container also offers non-functional services related to lifecycle (e.g. create, delete, etc.), bindings and invocations. A CCM application is built by defining an *assembly* entity using XML Schema templates.

Illustration: An avionics control application We exemplify our approach with a typical industrial avionics control application (figure 1, right) presented in [20]. In this application, a *Rate Generator* component sends periodic pulse events to a positioning sensor (GPS) component. This GPS refreshes cached coordinates available through a facet named *MyLocation*, and notifies a *Displaying device* with *Ready* events. This component reads current coordinates via its receptacle *GPSLocation*, and then updates the display.

3 Modelling and analysis of CCM-CBS

In order to allow qualitative and performance analysis of CBSs, we developed a general method, based on the *Stochastic Well-formed Model* (SWN), and concentrating on performance properties. An SWN [3] is a special class of coloured Petri net, a Well-formed (WN), with probabilistic extensions. The structured definition of an SWN allows us to compact its reachability graph in a *Symbolic Graph*, *SRG*, composed of *symbolic markings*. Moreover, the SRG provides an aggregate version of the underlying Markov chain of the SWN. To analyze a CBS, our method consists of two main phases: Generation of a global SWN of the CBS, called *G-SWN*, starting from the CBS description architecture and component implementations. Then, a structured analysis of the G-SWN is applied. The G-SWN of a CCM CBS is built by modelling first components and their interfaces with an SWN model termed a *Component SWN* or *C-SWN*. C-SWNs are then modified to be composable with others, in the sense of Petri net composition (fusion of places or transitions), leading to *Composable Component SWNs* (CC-SWNs). Finally, the interacting CC-SWNs are composed together through fusion of element interfaces, providing the G-SWN.

3.1 From CCM-CBS to SWN models

When modelling CCM based applications, we considered the following points:

- (i) We study “stable” (i.e. fixed) architectures of CCM-CBS and we do not address performance of reconfiguration behaviours, as performance indices are mainly computed over long periods (steady-state analysis). Hence, we do not model non-functional services pertaining to initialization and reconfiguration steps.
- (ii) We model explicitly the event channel, characterized in CCM, with only one publisher of a specific event type and several subscribers.
- (iii) We model container services not related to architecture modification, because lifecycle and binding services relate to transient configurations.
- (iv) We use basic colour classes to model data entities (requests and parameters, request or event data,...) and active entities (processes, threads).

In the sequel, before presenting component modelling, we describe the modelling of event interfaces. Details on facet/receptacle modelling can be found in our previous work [18] which defined mapping rules (1, 2 and 3) for translating requestor (receptacle) and service (facet) interfaces into the SWN context.

3.1.1 Event-based interfaces

Modelling event interfaces requires knowledge of the event handler implementation and of the resuming point after processing of event. An event handler may be implemented inside the subscriber(s) as an internal operation. It may also trigger a service request to another component which can be the publisher itself. This strategy, known as *control-push data-pull* mode, is commonly encountered when a data producer, being the publisher, publishes an event indicating that some data was updated and is ready to be consumed. To retrieve the data,

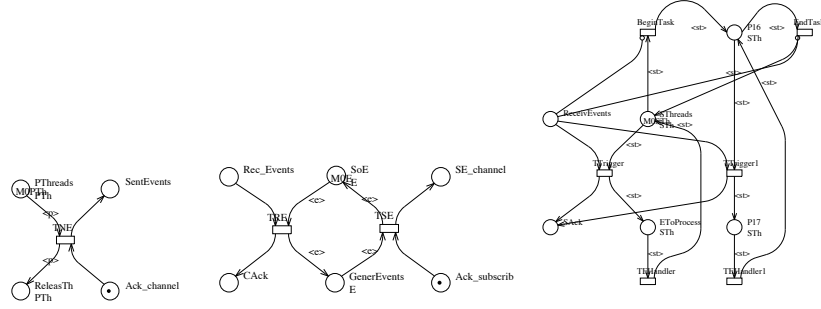


Figure 2: SWN models of event publisher interface (left), event channel (middle) and subscriber (right)

the subscriber calls an accessor method in a facet provided by the component storing the data. On the other hand, after an event processing, the subscriber may resume the suspended activity, or jump to another processing mode.

Because of space constraints, we present in the sequel the modelling of two cases: (i) the case of subscriber internal event processing and activity resuming, and (ii) the *control-push data-pull* case also with resuming activity.

The other cases can be deduced following a quite similar modelling.

Mapping rule 4: Event interfaces, case (i)(1)

- A publisher interface of a component, identified by a set *PTh* of colours modelling possible publisher threads, is modelled with a transition *TNE* representing the notification of events (see figure 2, left), with *Pthreads*, *Ack_channel* places as preconditions, and *ReleasTh*, *SentEvents* as postconditions.
- An event channel managing a set *E* of events of a specified type is modelled with two transitions *TRE* and *TSE*, expressing respectively receiving events from publishers and sending these events to subscribers (figure 2, middle). The *TRE* transition is controlled by *Rec_Events* and *SoE* places and has the *Cack* and *GenerEvents* places as postconditions. While the *TSE* transition is controlled by *GenerEvents* and *Ack_subscrib* places and has as postconditions the *SoE* and *SE_channel* places.
- The SWN of a subscriber interface of a component, identified by a set *STh* of colours modelling possible subscriber threads (see figure 2, right for an example), is made of two parts: (i) a local processing modelled by the *BeginTask* and *EndTask* transitions, controlled by two places *ReceivEvents* and *SThreads* places, and (ii) an event processing part triggered with the *TTrigger* transition modelling reception of an event. *TTrigger* is also controlled by *ReceivEvents* and *SThreads* and has *SAck* and *ETolProcess* as postconditions. The event handler is modelled with the *TEHandler* transition.

In the publisher model, places *Pthreads* and *Ack_channel* model respectively the publisher threads and the ready state of the component. Whereas, places *ReleasTh* and *SentEvents* model respectively the publisher threads resuming their activities and event notifications. The *Ack_channel* and *SentEvents* places are uncoloured as we model publishers notifying one specific type of events.

In the event channel model, the *Rec_Events* and *SoE* places model respectively received notifications and the set of possible events. *Rec_Events* is not

coloured. Postconditions of *TE* (*CAck* and *GenerEvents*) model respectively acknowledgment and generated events to be sent to subscribers. *CAck* is also not coloured, and *GenerEvents* has the same domain as *SoE*. The *Ack_subscribe* place models the “ready to broadcast events” state of the channel. It is also uncoloured as the *Ack_channel* of the publisher. The *SE_channel* place models events sent to subscribers by the channel.

In the subscriber model, when an event is received in place *ReceiveEvents*, the local processing is interrupted thanks to inhibitor arcs which prevent firings of transitions *BeginTask* and *EndTask* (and possibly others expressing other local activities). The *SThreads* place is coloured with the *STh* basic class, while *ReceiveEvents* has the same domain as the *SoE* place (neutral or set of event colours when dealing with several types of events). The reception of event causes the sending of an acknowledgment in the uncoloured place via the firing of the transition *TTrigger* (or *TTrigger1*), *SAck* and execution of an event handler. Note that we abstracted the model of event processing into one transition *TEHandler* (or *TEHandler1*). We can replace this transition with a subnet detailing the handler when we are interested in the impact of processing details on performances of the system. Globally, the triggering of the handler shall be achieved in a “short” period of time, with respect to components activities. This may be reflected in firing rates ratios (for instance 0.001/1.0) of *TTrigger*, *TTrigger1* transitions, over *BeginTask*, *EndTask* and *TEHandler*, *TEHandler1* transitions.

Dealing with multiple subscribers A publisher interface can push events to several subscribers. In this case, the C-SWN of the event channel must be modified (see mapping rule 5) in order to be composable at the same time with several publishers and subscribers models (fusion of places or transitions). This modification gives rise to a CC-SWN modelling the event channel. Note that CC-SWNs corresponding to publishers and subscribers components are the obtained C-SWNs without any modification. Moreover, in the figure, we model for clarity only one transition (*Task1*) for the local processing of the subscriber.

Mapping rule 5: Event interfaces, case (i)(2) The C-SWN of an event channel with multiple subscribers is modified by duplicating places *SE_channel*, *Ack_subscribe* and their arcs, as many times as there are subscribers.

Control-push data-pull mode In this case, the publisher is endowed with a facet and an event source, and the subscriber with a receptacle and an event sink. Mapping rule 7 gives the corresponding modelling. For ease of modelling, we abstract the local processing of the subscriber component to one transition.

Mapping rule 6: Event interfaces case(ii) The SWN of figure 3 models the control-push data-pull case. The event channel remains unchanged, as defined in mapping rule 4. The publisher, identified with a set *PTh* of colours modelling publisher threads, sends events through the *TNE* transition. It exposes a facet interface using a set *MP* of business methods parameterized with a set of parameters, and modelling the service providing data related to the event. The subscriber, identified with a set *STh* of subscriber threads, receives event notifications in the *Received Events* place, and triggers a service request through a receptacle (given by the transitions *TBRS* and *TERS*) to obtain event data.

In this modelling, the subscriber performs a local processing until receiving a notification of an event (in place *Received Events*). In this case, it sends an acknowledgment for this event through the transition *TTrigger*, and invokes a business method (transition *TBRS*) to the publisher component.

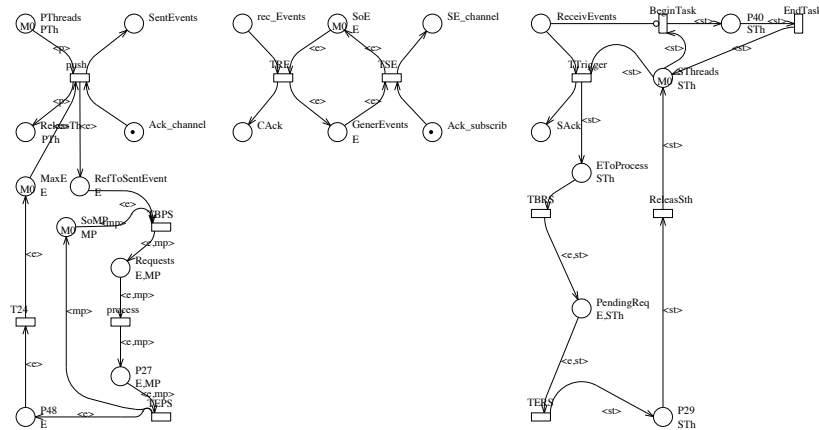


Figure 3: CC-SWNs of event interfaces in control-push data-pull mode

3.1.2 Building CC-SWNs for CCM components

Starting from each component implementation, the C-SWN of a component is built by analyzing the source code of the component, fixing a level of details for modelling and following the algorithm below:

CC-SWN BUILDING ALGORITHM

1. Translate each facet defined with the "provides" keyword using mapping rules 1 and 3. Model activities related to the service.
2. Translate each receptacle defined with the "uses" keyword using mapping rules 1 and 2.
3. Translate each event source defined with the "publishes" or "emits" keyword using mapping rules 4 or 6 and 5 eventually.
4. Translate each event sink defined with the "consumes" keyword using mapping rules 4, 6 and 5 eventually. Model activities related to the event handler.
5. If any processing involved in any interface invokes internal methods of the component, model activities associated with these functions.

Modelling activities in any stage is done by an expert. Obviously, abstraction may be used at this stage by selecting an appropriate level of details of components. At the highest abstraction level, an activity is modelled with a single transition. When we obtain the model, we associate rates to transition, as we use stochastic models. These rates may be estimated through a *model parameters estimation phase*, where a test application (see for instance the *Grinder* tool, <http://grinder.sourceforge.net>) is ran in order to measure the parameters needed for performance prediction.

Let us illustrate the building of the CC-SWN of a CCM component with the *NavDisplay* component of the avionics control system. We start from the implementation code given below:

```
eventtype tick {public rateHz rate};interface position {long get_pos()};
interface tickConsumer:Components:: EventConsumerBase
    { void push_tick(in tick the_tick)};
interface NavDisplay : Components:: CCMObject
```

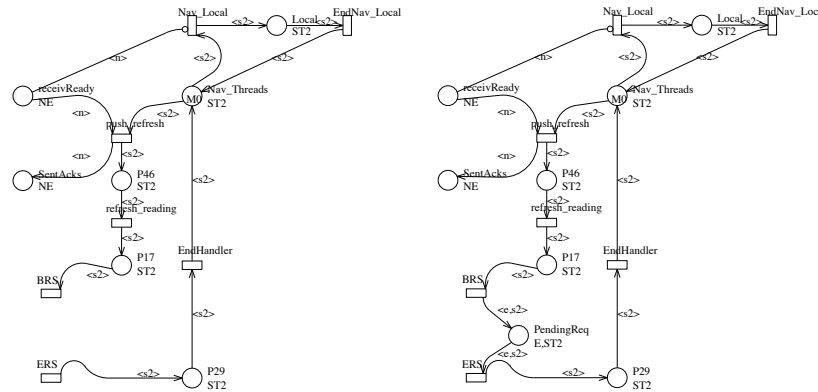


Figure 4: The NavDisplay C-SWN and CC-SWN

```
{ void connect_GPSLocation(in position c);
  position disconnect_GPSLocation(); tickConsumer get_consumer_Refresh();
  position get_connection_GPSLocation(); };
component NavDisplay {uses position GPSLocation; consumes tick Refresh;};
class NavDisplay_Executor_Impl: public virtual CCM_NavDisplay,
public virtual CORBA:: LocalObject
{ public: virtual void push_Refresh(tick *ev) {this->refresh_reading();}
  virtual void refresh_reading(void)
  { position_var cur = this->context->get_connection_GPS Location();
    long coord = cur->get_pos(); }; };

```

From the definition of the component interfaces, we derive first a model of the receptacle *GPSLocation*, with the *BRS* and *ERS* transitions. *ST2* is the basic colour class modelling the NavDisplay threads. We also model the event sink to which an event handler *refresh_reading* is associated, triggered by the GPS through the *push_refresh* operation. We model the handler with several transitions: a beginning transition *refresh_reading*, a couple of receptacle transitions *BRS*, *ERS* modelling the request to the GPS facet, and ending transition *EndHandler*. We obtain the C-SWN of figure 4, top. This C-SWN is completed using mapping rule 2, getting thus the CC-SWN of figure 4, bottom. We model the other components in the same way. Communication between the GPS and NavDisplay components is a control-push data pull scenario. Event channels are also modelled between the RateGen and GPS components, and the GPS and NavDisplay components as described in mapping rule 5.

3.1.3 Containers

A container is made up of a set of interconnected CCM components and a set of services offered to its components. It mediates invocations of components from or to external components belonging to other containers, through: (i) either a *callback* (external) interface which acts as an interceptor for all incoming calls to the component, (ii) or an interceptor for outgoing calls, internal to the container. So, building the SWN of the container requires connecting its components CC-SWNs, modelling the services and modelling the mediation role.

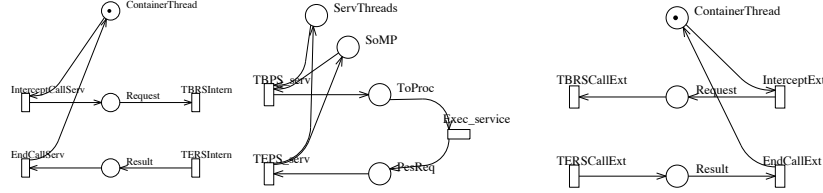


Figure 5: Modelling routing of a container service request (left), the container service (middle) and callback interfaces (right)

Connecting the components CC-SWNs Interconnection of the CC-SWNs is translated into the fusion of transition/places corresponding to interfaces of communicating components. Precisely, we proceed to:

- Fusion of transitions (TBRS, TBPS) and (TERS, TEPS) associated with facet/receptacles of each couple of interacting components.
- Fusion of places (SentEvents, Rec_Events) and (Ack_channel, CAck) associated to event interfaces of each publisher and its event channel, and places (SE_channel, ReceivEvents) and (Ack_subscriber, SAck) associated to event interfaces of each subscriber and the corresponding event channel.

Fusion of two transitions (resp. places) consists in defining a unique transition (resp. place) and keeping associated arcs of fused transitions (resp. places). Colour classes of the two transitions are mapped in one to one correspondence for common parameters of the interface and specific colour classes of each transition (resp. place) are kept. Hence, the colour domain of the resulting transition is the Cartesian product of colour classes of the fused transitions without repetition. Whereas, colour classes of two fused places are mapped in one to one correspondence leading to the colour domain of the fused place.

Modelling container services Achieving this modelling requires to associate a CC-SWN model to each container service and a second SWN modelling the routing of a component request to the service needed. We assume a monothreaded container. Mapping rule 7 describes this concern.

Mapping rule 7: Container services

- A container service is modelled with an abstracted component having one service interface, identified by a set of server threads colours and offering a set MP of methods (figure 5, middle). One transition *Execute* abstracts the service.
- Routing a request to an invoked container service is modelled with the model of figure 5, left. A single place *ContainerThread* models the unique thread of the container. The *InterceptCallServ* transition expresses intercepting a request. It is controlled by the *ContainerThread* place. The *TBRSEntern* transition models the request made by the container to its service. The result is obtained with the *TERSEntern* transition and sent to the requester using *EndCallServ* transition.

We choose to abstract the activity induced by a container service, as our goal is to consider the impact of a service on the execution of the analyzed CBS. On another side, the container can manage its component instances, threads or resources using pooling technique to reduce some overhead. If the designer is interested in knowing the impact of this pooling on performances of his application, we can consider this by associating a consequent rate to the transition

TBRSEnter related to the invoked operation. Note that, for more clarity, the SWNs given in figure 5 are not coloured, but should be.

Modelling the mediation role Outgoing component calls are mediated by the container. This is modelled, as in the case of a container service invocation, using mapping rule 7. External invocations to components services located inside a container are also intercepted by this container on an external interface, and routed to the concerned component. This is modelled with mapping rule 8.

Mapping rule 8: Callback interfaces

Routing an external invocation to a component service of a container is modelled with the model of figure 5, right. The *InterceptExt* transition models the interception of a request. It is controlled by the *ContainerThread* place. The *TBRSCallExt* transition represents the submission of a request to the concerned component. The result is obtained with the *TERSCallExt* transition and sent to the requester using *EndCallExt* transition.

CC-SWN of a container modelling a container leads to a CC-SWN whose interfaces are defined as the callback interfaces and non-connected interfaces associated with internal invocation of external services.

CONTAINER CC-SWN BUILDING ALGORITHM

1. Model each container service using mapping rule 7.
 2. Connect communicating CCM components.
 3. For each invocation of a container service, build a mediation part using mapping rule 7, connect it to the requested component in the left side, and to the service model in the right side.
 4. Model each callback interface using mapping rule 8, and connect it to the requested component service.
 5. For each internal invocation of a service offered by an external component, build a mediation part following mapping rule 7, and connect it to the requester component.
- End

3.1.4 Modelling the CCM application

Modelling a CCM application requires interconnection of its containers CC-SWNs. This is done via connection of their interfaces, as done for CCM components. The resulting CC-SWN is then completed, as usual, by “closing” interfaces of the application with a simple Petri net to provide a G-SWN with *finite* state space.

For our application, the three components are included in one container. Their CC-SWNs are interconnected. We then close the *rate_control* interface of the application (initially of the *RateGen* component), which provides the *start* and *stop* methods.

3.2 Structured performance analysis of CBS

After generating the G-SWN of a CCM-CBS and the $(CCSWN_k)_{k \in K}$ of the components, we apply the last step of our analysis approach, aiming mainly at computing performance indices of a system. We can also check qualitative properties like deadlocks or reachability of a particular state (i.e. a marking). Analysis of a CCM-CBS can be performed through the direct analysis of the G-SWN obtained in the first step of our method. This approach has been followed in [2] for analysis of a composition of SWNs, and implemented in the Algebra tool of the GreatSPN package [16]. In our approach, we rather benefit from the

Transition	Rate	Transition	Rate	Transition	Rate
NewTick	0.5	push_ready	0.75	return_location	0.9
push_pulse	0.8	push_refresh	0.8	BPS	8
push_tick	0.8	GPS_Local	0.4	Nav_Local	0.4

Table 1: Transition rates of the studied configuration

compositionality of a CBS in order to provide an efficient steady-state performance analysis with regard to computation time and memory requirements.

For this purpose, we devise an extension of our previous work [5, 6] adapted to CBS, and applied here to analysis of CCM-CBS. Let us remind that our previous approach defines a structured analysis method which decomposes a (global) SWN into several subnets connected in either a synchronous or else an asynchronous manner, and study each subnet augmented with “parts” abstracting interactions with other subnets. These separated studies are used to derive a tensorial representation of the generator of the underlying aggregated Markov chain of the global net, used to compute performance indices.

3.2.1 Structured analysis method for CBS, applied to CCM-CBS

Extension to CBS of our structured analysis rises three problems:

1. Composition of CC-SWNs of components, as we start from the definition of components in the case of a CBS. This is in contrast to the previous method where a global SWN is decomposed into several subnets. Composition of SWNs models of a CBS has been explained above.

2. Bringing an interconnection of components into a synchronous or an asynchronous composition of SWNs. We map a request/response interaction into a synchronous composition of CC-SWNs, while we model an event interaction with an asynchronous composition of CC-SWNs. We emphasize here that our modelling of interfaces *ensures* that conditions of synchronous and asynchronous compositions of subnets given in [12, 11] are fulfilled.

3. Impact of having mixed synchronous and asynchronous compositions in the same global model, as the structured method was defined for either a synchronous composition or else asynchronous composition of SWNs. This problem requires the following sufficient conditions for applying our method:

- (i) If $(\mathcal{N}_1, \mathcal{N}_2)$ and $(\mathcal{N}_1, \mathcal{N}_3)$ (resp. $(\mathcal{N}_2, \mathcal{N}_3)$) are in pairwise client/server relationship, then $(\mathcal{N}_2, \mathcal{N}_3)$ (resp. $(\mathcal{N}_1, \mathcal{N}_3)$) are not in client/server relationship.
- (ii) If $(\mathcal{N}_1, \mathcal{N}_2)$ are in publish/subscribe relationship and in client/server relationship too, then event colours are not involved in the client/server interaction.
- (iii) If $(\mathcal{N}_1, \mathcal{N}_2)$ and $(\mathcal{N}_1, \mathcal{N}_3)$ are in publish/subscribe relationship, and if $(\mathcal{N}_2, \mathcal{N}_3)$ are in client/server relationship, then event colours are not involved in the $(\mathcal{N}_2, \mathcal{N}_3)$ interaction.

We give next our analysis algorithm based on the structured method. We start with the G-SWN of the application and the set of CC-SWNs corresponding to components $E = \{\text{CC-SWN}_k \mid 1 \leq k \leq K\}$.

1. Find the set of SWN subnets $(\mathcal{N}_k)_{1 \leq k \leq K'}$ representing a possible decomposition of the G-SWN, that fulfill conditions stated in [12, 11] for a structured representation of the SRG and its aggregated generator. These SWNs do not necessarily correspond to the CC-SWNs of the set E due to restricted conditions above. This point is investigated by checking first service invocation interac-

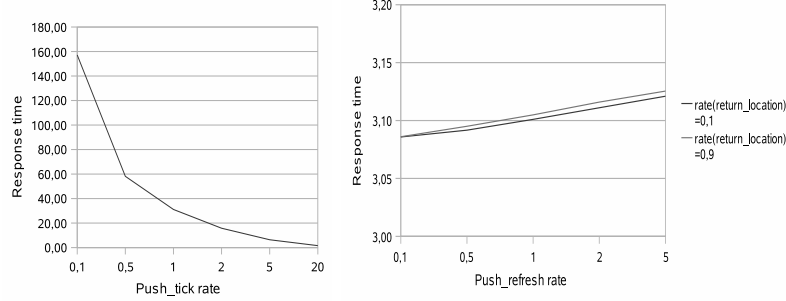


Figure 6: GPS processing request response time (left) and NavDisplay event handling response time (right) with respect to component receiving rate

tions, and then event interactions.

2. Extension of the SWNs \mathcal{N}_k to autonomous (said *extended*) SWNs $\tilde{\mathcal{N}}_k$.
3. Generation of the SRGs of these extended SWNs.
4. Computation of the synchronized product of these SRGs and of the tensorial representation of the generator of the underlying aggregated Markov chain.
5. Computation of the steady state distribution of the aggregated model and computation of the required performance indices.
6. Expression of the results in the initial context of the components.

Automation of points 1 and 6 are currently under development, whereas the the others steps have been automated in a tool *compSWN*, the new version of the TenSWN tool [5].

4 Illustration

The set of (CC-SWN $_k$) obtained when modeling components satisfy conditions for a structured analysis. We use our tool *compSWN* on this set to compute steady-state probabilities. The solver runs on a Suse linux 9.2 workstation with Intel Pentium IV (3GHz) and 512 MO.

We are interested in studying the variation of two performance indices: (i) the response time of processing a request in the GPS, with regard to its notification receiving rate (*push_tick* rate), and (ii) the response time of the event handling in the NavDisplay, with respect to its notification receiving rate (*push_refresh* rate). We choose a configuration of the CBS leading to an SRG size of 281760 symbolic markings (3937280 ordinary markings). We take fixed rate values of a critical set of transitions (see table 1); then, we vary transition rates (not mentioned transitions have rate equal to 1, i.e. faster than all others, rates being given in the same unit). We obtain diagrams of figure 6.

The left diagram shows an improved response time as the receiving rate of the transition *push_tick* increases. This is somewhat surprising, as the response time gets reduced when the GPS gets overloaded. This a priori contradictory behaviour indicates that the system is not stable in the initialization phase, but adapts its activity to events arrival. The right diagram for two different

request processing rates in the GPS (*return_location* transition) shows a very slightly increasing response time with increasing of event receiving rate in the NavDisplay. The increasing of response time is expected since the load of the two components (NavDisplay and GPS which processes the event request) becomes more important. However, we note the very slow increasing of the response time, which shows a good configuration not saturated for a long period of time.

5 Conclusion

In this paper, we have proposed a method allowing us to study, in an efficient way, performances of Component Based Systems designed with the CORBA Component Model. The approach starts from the architecture description of a CCM based application and the code of its components. It provides an SWN (the G-SWN) of the whole system and a collection of SWNs corresponding to the components or the containers of the application. A structured compositional analysis method is used to compute performance indices. If, SWNs of the components are complex, this approach provides memory and time savings. Current work relates to automatizing information extraction for direct CCM interface. Future work will try to extend application conditions for structured analysis and semi-automatic verification of these conditions.

References

- [1] T. Barros, A. Cansado, E. Madelaine, and M. Rivera. Model checking distributed components: The Vercors platform. In *FACS. ENTCS*, 2006.
- [2] S. Bernardi, S. Donatelli, and A. Horváth. Implementing compositionality for stochastic Petri nets. *Int. J. STTT*, (3):417–430, 2001.
- [3] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed colored nets and symmetric modeling applications. *IEEE Trans. on Comp.*, 42(11):1343–1360, Nov 1993.
- [4] C.U.Smith and L.G.Williams. *Performance Solutions*. A.-Wesley, 2002.
- [5] C. Delamare, Y. Gardan, and P. Moreaux. Efficient implementation for performance evaluation of synchronous decomposition of high level stochastic Petri nets. In *ICALP2003*, pages 164–183, Holland, 2003.
- [6] C. Delamare, Y. Gardan, and P. Moreaux. Performance evaluation with asynchronously decomposable SWN: implementation and case study. In *PNPM03*, pages 20–29, USA, 2003. IEEE.
- [7] L. Dias da Silva and A. Perkusich. Composition of software artifacts modelled using colored Petri nets. *SCP journal*, 56(1-2):171–189, 2005.
- [8] D.Petriu, C.Shousha, and A.Jalnapurkar. Architecture-based performance analysis applied to a telecommunication system. *IEEE Transactions on Software Engineering*, 26(11):1049–1065, 2000.

- [9] E.Weyuker and F.Vokolos. Experience with performance testing of software systems: issues, an approach and case study. *IEEE Transactions on Software Engineering*, 26(12):1147–1156, 2000.
- [10] V. Grassi, R. Mirandola, and A. Sabetta. Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach. *J. Syst. Softw.*, 80(4):528–558, 2007.
- [11] S. Haddad and P. Moreaux. Asynchronous composition of high level Petri nets: a quantitative approach. In *ICATPN'96*, volume 1091, pages 193–211.
- [12] S. Haddad and P. Moreaux. Aggregation and decomposition for performance evaluation of synchronous product of high level Petri nets. 1996.
- [13] J.M. Ilić, S. Baarir, M. Beccuti, C. Delamare, S. Donatelli, C. Dutheillet, G. Franceschinis, R. Gaeta, and P. Moreaux. Extended SWN solvers in GreatSPN. In *QEST 2004*. IEEE Comp. Soc. Press., 2004.
- [14] N. Medvidović and R. N. Taylor. A classification and comparison framework for software architecture description languages. In *IEEE Trans. On Soft. Eng.*, volume 26, pages 70–93, 2000.
- [15] OMG. CORBA component model specification. version 4.0. <http://www.omg.org/cgi-bin/apps/doc?formal/06-04-01.pdf>, 2006.
- [16] Perf. Eval. Group. GreatSPN home page: <http://www.di.unito.it/~greatspn>, 2002.
- [17] A.E. Rugina and al. A system dependability modeling framework using AADL and GSPNs. Technical Report 05666, LAAS, 2006.
- [18] N. Salmi, P. Moreaux, and M. Ioualalen. Formal models of fractal component based systems for performance analysis. In RNTI-SM-1, editor, *ISoLA 2007*, pages 49–60, Poitiers, France.
- [19] C. Szyperski, D. Gruntz, and S. Murer. *Component Software - Beyond Object-Oriented Programming*. Addison Wesley - ACM Press, 2002.
- [20] N. Wang and C. Rodrigues. Tutorial on corba component model (CCM). BBN Technologies and Washington University, July 6th 2003.
- [21] X. Wu and M. Woodside. Performance modeling from software components. *SIGSOFT Softw. Eng. Notes*, 29(1):290–301, 2004.

Mapping WSLA on Reward Constructs in Möbius

Rouaa Yassin Kassab* Aad van Moorsel†

Abstract

This paper provides an outline for generating Möbius reward constructs and a partial Stochastic Activity Network model from an existing Web Service Level Agreement (WSLA) document of a web service system. This is done by parsing the WSLA document to group its ‘Service Level Agreement parameter elements’ with their corresponding ‘metric elements’ using the Document Object Model parser. These groups then are represented by matching Stochastic Activity Network representations and reward C++ statements that can be integrated in the Möbius framework. Using this approach, service providers can predict SLA values by solving the produced model.

1 Introduction

The rapidly increasing popularity of the service provider paradigm raises the necessity to define clear relationships between service providers and their potential customers regarding the offered Quality of Service (QoS) (like response time, availability, etc) [1, 2]. Service Level Agreements (SLA) set the rules of this relationship and specify additional information such as the penalties that should be paid in case of the contract breaching [3].

To specify the different service levels, specification languages such as Web Service Level Agreement language (WSLA) [4] have been developed. WSLA is based on XML [5] and is used to describe SLA parameters of a web service and the way to compute their values [6]. If SLAs are not met, the provider typically pays a penalty [7]. Therefore, it is of interest to model and predict if systems meet SLAs.

We are not aware of any special modelling tool that provides a model for a system given its SLA. Therefore in this paper, we develop a way to create this model from a given WSLA document using the Möbius framework components [8]. This is done by using the Document Object Model parser (XML DOM) [9] to examine the WSLA document and group each SLA parameter with the elements that compute its value. It is worth mentioning that we have found that it is not possible to map SLA elements to the Möbius reward constructs

*School of Computing Science, Newcastle University, UK rouaa.yassin-kassab@ncl.ac.uk

†School of Computing Science, Newcastle University, UK aad.vanmoorsel@ncl.ac.uk, supported in part by UK DTI P0007E (‘Trust Economics’), UK EPSRC EP/D037743/1 (‘Networked Computing in Inter-Organisation Settings’), EU network of excellence 026764 (‘ReSIST: Resilience for Survivability in IST’) and EU coordination action 216295 (‘AMBER: Assessing, Measuring, and Benchmarking Resilience’).

only. This is caused by dependence between WSLA elements and the inability to store a set of the reward values generated over an interval of time. Therefore, we represent each element in the parameter group with either a set of Stochastic Activity Network (SAN) primitives or a set of C++ statements of the Möbius reward model function, or a combination of these two.

This paper is organized as follows. In Section 2, we provide the background of the WSLA language and the Möbius modelling tool. The problem we address is then described in Section 3. A solution is proposed and discussed using a running example in Section 4. Finally, the conclusion is drawn and future work is pointed out.

2 Background

2.1 WSLA Language

The primary role of the WSLA language is to formalize service level agreements in order to automate the configuration of the service provision system and the QoS monitoring system [4]. To achieve this, the WSLA specification contains three important constructs:

- Parties: describes the players that are involved in the SLA contract.
- Service Definition: contains a comprehensive definition of SLA parameters and how to use metrics to compute their values. Each metric will be measured from a source by identifying a measurement directive. In the case of a composite metric, its value will be accumulated using a function (that takes other metrics or constants as its operands). We are particularly interested in this part because it contains the SLA parameter descriptions.
- Obligations: includes Service Level Objectives which are the agreed values of SLA parameters in a specified duration, and what action should be taken in the case of contract violation.

WSLA language rules are designed to be thin. However, to cover the complexity of real systems, WSLA allows one to create new types which express domain specific SLA concepts [4]. In this paper, we limit our attention to the WSLA core, which provide the most important requirements for the system providers, and its standard extensions functions and measurement directives. We do not consider constructs that could be extended according to the domain specific needs (like using different measurement directives and functions depending on Web Service Definition Language defined operation) [4]. Furthermore, in the WSLA Metric definition, we depend on measurement directive and functions to compute the metric values. We ignore both the measurement directive variable element and the MetricURI element which define the location from which metric value can be taken directly.

2.2 Möbius

The Möbius tool is used in modelling the performance of a wide range of computing systems [8]. It is a framework that comprises multiple modelling formalisms

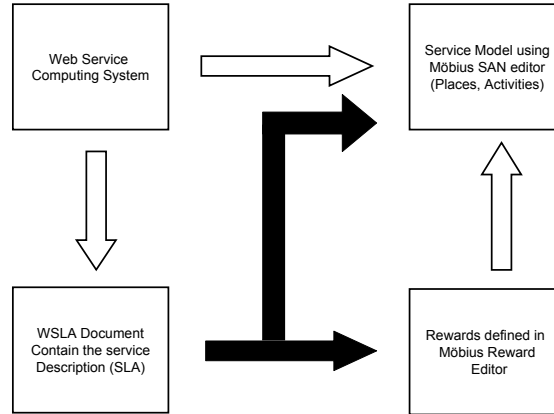


Figure 1: Problem Description

(SAN, Bucket and Balls, PEPA, Fault tree) and multiple solution methods (simulation, numerical solvers), and many of its methods are independent and can be used in combination with each other [8]. To measure the performance attributes of a given system using Möbius, we should [8]:

- Build an atomic model that depicts all the relevant system states (using tokens in the simple or extended places). State changes occur by actions (timed or instantaneous activities) which fire according to a distribution rate (deterministic, exponential, etc). Input/Output Gates can also be defined to represent the enabling predicate of an activity or the marking change of a place.
- Create the composite model if necessary (when the atomic model is a part of a larger model).
- Define the rewards from which metrics will be computed. There are two kinds of rewards; rate rewards represent the time spent in each state (place) and impulse rewards count activity completions. Each reward variable has a reward function that computes its value, and time type that specifies when the reward function should be evaluated.
- Solve the model by using either simulation or numerical solvers.

3 Problem Description

In Figure 1, we graphically depict the problem we tackle. It indicates a web service with an SLA specified by a Web Service Level Agreement document. Independently, this web service is modelled using Stochastic Activity Networks

in the Möbius framework. Rewards are defined on this model using the reward structures in Möbius.

Let us assume that we only have the WSLA document for a given service. The objective of our work is to automatically generate a complete reward definition in Möbius and the part of the SAN model (places and activities) that will be needed to accomplish the reward description. Black arrows in Figure 1 indicate the problem we address.

```

<Schedule name="hourlyschedule">
  <Period>
    <Start>2001-11-30T14:00:00.000-05:00</Start>
    <End>2001-12-31T14:00:00.000-05:00</End>
  </Period>
  <Interval>
    <Minutes>60</Minutes>
  </Interval>
</Schedule>

<SLAParameter name="TransactionRate" type="float" unit="transactions/hour">
  <Metric>Transactions</Metric>
  <Communication>
    <Source>ACMEProvider</Source>
    <Pull>ZAuditing</Pull>
    <Push>ZAuditing</Push>
  </Communication>
</SLAParameter>

<Metric name="Transactions" type="long" unit="transactions">
  <Source>ACMEProvider</Source>
  <Function xsi:type="Minus" resultType="double">
    <Operand>
      <Function xsi:type="TSSelect" resultType="long">
        <Operand>
          <Metric>SumTransactionTimeSeries</Metric>
        </Operand>
        <Element>0</Element>
      </Function>
    </Operand>
    <Operand>
      <Function xsi:type="TSSelect" resultType="long">
        <Operand>
          <Metric>SumTransactionTimeSeries</Metric>
        </Operand>
        <Element>-1</Element>
      </Function>
    </Operand>
  </Function>
</Metric>

```

```
<Metric name="SumTransactionTimeSeries" type="TS" unit="transactions">
  <Source>ACMEProvider</Source>
  <Function xsi:type="TSConstructor" resultType="TS">
    <Schedule>hourlyschedule</Schedule>
    <Metric>SumTransactions</Metric>
    <Window>2</Window>
  </Function>
</Metric>

<Metric name="SumTransactions" type="long" unit="tansactions">
  <Source>ACMEProvider</Source>
  <MeasurementDirective xsi:type="InvocationCount" resultType="long"/>
</Metric>
```

Table 1: WSLA Example.

It is important to note that it is not possible to model SLA parameters into Möbius just using rewards. We therefore also map WSLA partly on a SAN, essentially to model additional reward constructs. This is for two reasons:

1. Möbius does not accept to use the value of a reward variable (that is generated by solving the model) as an input to another reward variable in the same model [8]. This means that reward variables in the same model in Möbius cannot depend on each other. However, in WSLA, the value of SLA parameter depends on the value of a metric that may depend on another metric and so on. To work around this, we use a SAN model, which represents a part of the required metric, to be used in the computation of SLA parameters.
2. There is no ability to store a set of reward values during a period of time in Möbius. However, in the computation of most SLAs, WSLA depends on a set of metric values that is taken through a defined period of time or series of events. Therefore, we use extended places in SAN, allowing us to store values according to the firing of an activity.

In WSLA, as we mentioned before, an SLA parameter depend on metrics to compute its value [4]. These metrics could be either resource metrics, which use a measurement directive to measure its value, or composite metrics, which depend on the value of other resource metrics through a specified function [4]. Therefore, we first provide the mapping for resource metrics.

Every SLA should at least depend on a metric that uses the measurement directive. That is because if SLA parameter depends on only one metric, it should be a resource metric. To illustrate this further, see the example in Table 1, which was taken from the IBM WSLA language specification [4]. It specifies an SLA which reflects the rate of an operation invocation. The value of this SLA is derived by computing the difference between the last two measures of the invocation values which are taken on an hourly basis.

This example computes the value of the SLA parameter **TransactionRate** depending on the use of the resource metric **SumTransactions**, and the com-

posite metrics **SumTransactionTimeSeries** and **Transactions**. The value of the metric **SumTransactions** should be produced first by using the measurement directive **InvocationCount**. The result then will be input to the function **TSConstruktor** to compute the value of the **SumTransactionTimeSeries** metric. After that, the last two values of the previous metric will be derived by the use of two copies of the function **TSSelect** in the third metric **Transactions**. Finally, these values are used as the operands of the function **Minus** in the same metric to produce its final value. This will be used then to generate the value of the SLA parameter.

4 Solution

The following steps establish the mapping from WSLA on Möbius rewards:

1. We start by grouping each SLA parameter that appears in the WSLA document with the metrics that compute its value.
2. For each group, we work from the metric that contains the measurement directive element up to the SLA parameter element (passing through metrics that use function elements to compute their values).
3. After mapping each SLA element, metric element and function element to its corresponding representation in Möbius, we will represent this in a new XML based document that can be input into Möbius.

We implemented the first step in the above approach using the Document Object Model parser (XML DOM). The third step is beyond the scope of this paper and is future work. Here we will focus only on the second step because it is the most sophisticated phase in mapping from WSLA to Möbius. For that we try to find and explain the representations for WSLA function and measurements directive elements through the running example of Table 1 in Section 4.1 and 4.2. Because of space limitations, we are not able to provide the solution for all the WSLA elements.

4.1 Measurement Directive

A measurement Directive element is used when the value of a Metric should be measured directly from a resource by probing or instrumentation of the system [8]. There are seven types of measurement directives in the WSLA specification. We represent each of them through one or more places in the Möbius Atomic model editor. This place then will be used as the basis to build the functions on.

Referring to Table 1 with the WSLA example, we see that the value of the SLA parameter **InvocationRate** is computed using three Metrics **SumTransactions**, **SumTransactionTimeSeries** and **Transactions**. The first one will take the reading of the current operation's invocation value to be stored by the second metric in an hourly schedule. The last metric then will subtract the last two values to obtain the transaction rate. Note that there is only one measurement directive **InvocationCount** and it is used by the last metric **SumTransitions**. We map this in Möbius as depicted in Figure 2, using the following SAN primitives:

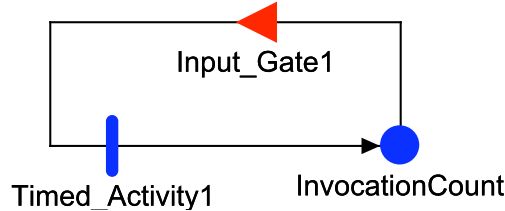


Figure 2: The representation of InvocationCount element in Möbius

- **InvocationCount** simple place: stores the number of operation invocations in the modelled system as the number of tokens in this place.
- **Timed_Activity1**: connects to the place to increase its marking according to an invocation rate.
- **Input_gate1**: controls the enabling of the activity and limits its firing times to not go to infinity. This is necessary in order to be able to solve the model.

By the use of this representation, we could have continuous readings of the value of the operation invocations. Thus, we could substitute the work of the **SumTransition** metric in WSLA example.

4.2 Functions

Function elements are used for a Metric if the Metric value is derived from the value of other metrics or constants [8]. There are eighteen types of functions used in WSLA document. Some of them will be mapped to the Möbius Atomic model editor, while the others are mapped to the reward function.

4.2.1 Functions that are mapped to the Atomic model editor

In WSLA, there are two functions (**TSConstructor** and **QConstructor**) which will be mapped to SAN primitives. Those functions are used to store a number of values, which are taken from another metric or function through a specific period of time, to be used by other metrics. Recalling the example again, and moving to the next metric, we see that the value of the **SumTransition** metric is used by the composite metric **SumTransitionTimeSeries** which use the **TSConstructor** function. This function will be mapped as follows (See Figure 3):

- **SumTransactionTimeSeries** extended place: It stores an array of values that are taken from the metric that is specified in the **Metric** element in the **TSConstructor** function (the **SumTransaction** metric that is represented by the **InvocationCount** place in our example). The number of



Figure 3: The representation of TSConstructor in Möbius

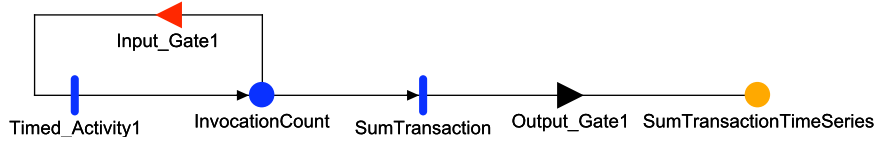


Figure 4: The produced atomic model

elements in the array is equal to 2, which is the value of the `<window>` element in the `TSConstructor` function.

- **SumTransaction** activity: The role of this activity is to specify the times that a new value will be added to the **SumTransactionTimeSeries** extended place. This is done by using a deterministic firing rate equal to the value that was given in the **Interval** element of the **Schedule** element in the `TSConstructor` function (60 minutes in our example).
- **Output_Gate1**: This is used to store each incoming value from the **SumTransactions** Metric in the corresponding array element in the **SumTransactionTimeSeries** extended place. The index of the array will start from 0 to its maximum value (array size minus one), returning the index to zero when the number of values reaches the array index maximum value.

Here, we could store a series of a particular size for the readings copies that was taken from the **SumTransition** metric according to a predefined schedule of time. So, we could replace the use of the **SumTransactionTimeSeries** metric.

After joining the previous two graphs that appear in Figure 2 and Figure 3, we will have the graph that is depicted in Figure 4.

By using this SAN graph, we could obtain the value of the number of the operation invocations according to an hourly schedule, and retain the last two.

Place Name	InvocationCount	SumTransactionTimeSeries
Initial Markings	0	0

Timed Activity	SumTransaction	Timed Activity1
Distribution Parameters	Value:60	Rate:1
Activation Predicate	(none)	(none)
Reactivation Predicate	(none)	(none)

Input Gate	Input Gate1
Predicate	InvocationCount -> Mark() < 100
Function	;
Output Gate	Output Gate1
Function	if (I < 1) { I =I+1; //where I is a global variable SumTransactionTimeSeries->Index(I)->Mark() = InvocationCount->Mark(); } else { I=0; SumTransactionTimeSeries->Index(I)->Mark() = InvocationCount->Mark(); }

Performance Variable Model: Invo reward		
Top Level Model Information	Child Model Name	example
	Model Type	SAN Model

Performance Variable : TransactionRate		
Affecting Models	example	
Reward Function	(Reward is over all Available Models) return((example->SumTransactionTimeSeries->Index(1)->Mark()) - (example->SumTransactionTimeSeries->Index(0)->Mark()));	
Simulator Statistics	Type	Steady State
	Options	Estimate Mean
		Include Lower Bound on Interval Estimate
		Include Upper Bound on Interval Estimate
		Estimate out of Range Probabilities
		Confidence Level is Relative
	Parameters	Initial Transient
		0.0
	Confidence	Batch Size
		1.0
		Confidence Level
		0.95
		Confidence Interval
		0.1

Table 2: The produced Möbius reward functions.

4.2.2 Functions that are mapped to the reward function coding

The remaining functions will be represented in the reward function in Möbius reward model. Since C++ is used to write the code for different Möbius components (functions, etc.) [8], all the functions' representation will be encoded in C++ as well.

As Möbius reward model has a time type through which its function will be

computed, the `<Interval>` element of the `<schedule>` element for each of the WSLA functions (of this type) will be mapped into this time type.

In our example, the last thing to do is to subtract the last two values that are stored in the `SumTransactionTimeSeries` metric.

We see that the composite metric `Transactions` uses the function `Minus`, which has two operands. These operands are the last two values of `SumTransactionTimeSeries` metric (represented in the `SumTransactionTimeSeries` extended place) that was taken through the use of the `TSSelect` function. The `TSSelect` function will be used as a part of the reward function in order to be able to choose the element with a certain index that is specified in the `<element>` element in the `TSSelect` function. The `Minus` function will also be included in the reward function to compute the subtraction between the two values.

```
Return ((SumTransactionTimeSeries->Index(1)->Mark ()) -
(SumTransactionTimeSeries->Index(0)->Mark ()));
```

A `Return` statement is used here identical to the Möbius reward encoding.

If we put the previous representation in the Möbius tool, Table 2 states the parameter values and the coding for each primitive and for the reward function.

5 Conclusion

This paper provided a mapping of SLA parameter elements in WSLA on Möbius reward and atomic models. It focused mainly on finding a representation in Möbius for WSLA function and measurement directive elements. In future work we implement this in Möbius. We are also developing generalisations of this approach to map WSLA to other reward structures than those used in Möbius.

References

- [1] Liu, Z., M.S. Squillante, and J.L. Wolf. On Maximizing Service-Level-Agreement Profits, in *Proceedings of the 3rd ACM conference on Electronic Commerce*, 2001.
- [2] Marilly, E., Martinot, O., Betg-Brezetz, S., and Delgue, G., Requirement for Service Level Agreement Management. IP Operations and Management, 2002 IEEE Workshop 2002: p. 57-62.
- [3] Jin, L.J, V. Machiraju, and A. Sahai, Analysis on Service Level Agreement of Web Services, in HP Labs Technical Reports. 2002, HPLaboratories.
- [4] Ludwig, H., Keller, A., Dan, A., King, R.P., and Franck, R. Web Service Level Agreement (WSLA) Language Specification, Version 1.0. (January 2003), IBM Corporation
- [5] W3C, XML Schema Definition Language.
- [6] Keller, A. and H. Ludwig, The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, March 2003. 11(1): p. 57-81.

- [7] Wolter, K. and A.v. Moorsel, The Relationship between Quality of Service and Business Metrics: Monitoring, Notification and Optimization, in HP Labs Technical Reports. April, 2001, HP Laboratories.
- [8] Sanders, W.H., Möbius User Manual, Version 2.0 draft. 2006, University of Illinois
- [9] W3C. Document Object Model (DOM). <http://www.w3.org/DOM>.

Using Representative Intervals for Trace-Based Performance Analysis of Embedded Device Use Cases

Lukas Pustina* Simon Schwarzer† Peter Martini‡

Abstract

Especially, in the development of embedded devices like mobile phones, performance evaluation plays an important role. The processing power is limited due to space, heat, and power restrictions of these devices. Therefore, design alternatives and different architecture configurations have to be analysed as early as possible in the development cycle. A typical approach for the analysis of different architectures is the simulation of instruction traces of use cases captured on already existing platforms. The drawback of this approach is that for different inputs, new traces have to be generated which require real input data and capturing time. Our approach analyses the input sensitivity of typical embedded device use cases in a prerequisite step. For use cases with a low input sensitivity it is possible to store representative intervals of selected program executions in a database. These representative intervals in combination with rules which describe the influence of input parameters are used to predict the performance of future architectures. The approach is evaluated for GSM and JPEG algorithms for which the frame rate and the number of pixels are used as input parameters. Thus, developers have to provide the relevant input data parameters only and the runtime of the simulations is reduced.

1 Introduction

New application areas for embedded devices like multimedia phones and short product cycles of these devices make performance evaluation of new hardware platforms necessary already in early stages of development. In these development phases, hardware prototypes are not yet available. Thus, it is a common practise to use detailed processor simulations to analyse the performance of different platform architectures.

The performance evaluation of future platforms for embedded devices is use case driven and is not dominated by wide range benchmarks, because the application areas of the devices are known and limited in comparison to desktop computers. Typical use cases for modern multimedia phones are the compression

*Computer Science IV, University of Bonn, Germany, pustina@cs.uni-bonn.de

†Computer Science IV, University of Bonn, Germany, schwarzer@cs.uni-bonn.de

‡Computer Science IV, University of Bonn, Germany, martini@cs.uni-bonn.de

or decompression of JPEG pictures and encoding and decoding of GSM voice data on the multipurpose CPU instead of a specialised chip.

Nevertheless, simulations of these use cases are very time consuming, because all performance relevant aspects of the platform have to be considered and millions of instructions have to be simulated. A series of approaches to reduce the simulation time of processor simulations are presented in the literature. However, all approaches focus on the acceleration of given combinations of use case and input data.

This paper presents a method that utilises ideas of the acceleration approaches to identify the influence of input data on given use cases. Characteristics of the program executions are used to identify phases in the use cases. By analysing the changes in the occurrence of the phases, the influence of the input data is identified. This knowledge is then applied to predict the performance for general input data. This is done by projecting the simulation results of representative samples depending on given input data parameters of the use case. In this way, the method eases the analysis of new use case configurations and accelerates the evaluation of different architecture configurations. Developers have to provide relevant input data parameters only and do not need to provide real input data. The number of pixels of an image or the frame rate of an audio stream are examples for such input parameters.

The rest of the paper is structured as follows. Section 2 gives an overview of related work. Section 3 introduces our method and describes the analysis steps in more detail. An evaluation of the approach is presented in section 4. Section 5 concludes the paper and outlines future work.

2 Related Work

There are two types of approaches for the simulation of performance models of processors and platforms, i.e. execution-driven and trace-driven [JE06]. Execution-driven simulations process compiled programs as input and simulate their execution on the modelled platform. SimpleScalar [ALE02] is an example for this type of processor simulation. Trace-driven simulations focus only on timing-relevant aspects of the model and do not simulate the functionality. A stream of instructions captured during a former execution is used as input. Both approaches have pros and cons which are discussed in detail in [JE06]. The main drawback of the execution-driven approach is the necessity to port operating systems and drivers to the simulator environment. In contrast, the main advantage of the trace-driven approach is that no operating systems and drivers have to be ported, so that every program available as instruction trace may be simulated. On the other hand, a drawback of this approach is the missing possibility to simulate the consequences of branch predictions, because the necessary information is usually not included in the instruction traces. Since the missing branch prediction consideration has typically no significant influence on the accuracy of the simulation results [YEL⁺06], this drawback is negligible. Processor emulators like Qemu [Bel07] emulate the functionality of a processor in software, but do not take into account the timing of instructions or components of the architecture. This kind of programs may be used to capture instruction traces instead of special hardware is necessary to capture instruction traces at the processor level.

We use trace-driven simulations to predict the performance of embedded device architectures for given program executions already during early development stages when the system is not fully specified. Instruction traces gathered on existing platforms are used as input for the platform configurations under study to predict their performance. This approach is feasible because successor architectures usually remain backward compatible.

Since ARM processors are widely deployed in embedded devices like mobile phones, instruction traces captured at processors of the ARM family are used in the analysis. Models of these platforms include the performance relevant components and aspects, i.e. pipeline behaviour of instructions, caches, buses, memory system, and the interconnections of these components. Details of the used trace-driven simulator and the modelling of ARM9 and ARM11 platforms with UML have been presented in [PSM08].

The standard approach of platform performance evaluation for a given use case consists of two essential steps. First, the input data for the use case is specified. For example, in case of JPEG, the resolution of the pictures and the sample pictures itself are specified. Second, the performance of a platform for the specified use case is analysed by simulating the instruction trace on a model of the platform. Approaches that reduce the simulation time base on two basic ideas. First, the reduction of the total number of necessary simulations for the use case while preserving the accuracy. Second, the reduction of the runtime of single simulations.

Approaches for both ideas are presented in the literature [EVB03, SPHC02, SPC01, DCD03, WWFH03]. Reducing the number of necessary simulations is achieved by analysing characteristics of program executions for different inputs. By grouping executions depending on their architecture-independent characteristics, it is possible to simulate only one representative of each group. Eeckhout et al. examined the influence of different inputs on programs of the SPEC2000 benchmark [EVB03]. With the help of architecture-independent characteristics and cluster analysis inputs with similar executions are identified. Similar methods can be applied to intervals of a single program execution. In this way, it is possible to identify representative intervals for the whole execution of a particular program [SPHC02, SPC01, DCD03, WWFH03].

3 Methodology Description

All mentioned approaches focus on the acceleration of simulations with a fix combination of use case and input data. Therefore, the input data need to be provided by the developers. For example, if developers want to analyse the runtime of the compression of a picture with different resolutions, they have to provide a real picture files.

In contrast, our approach predicts the runtime of use cases on the basis of relevant parameters of the input only. Methods presented in the literature are applied to identify the start, end, and *parameter dependent* phases of program executions. By analysing the length of the parameter dependent phase for different inputs, it is possible to derive rules describing the influence of significant input parameters. These rules can be used to project the execution time for different use case inputs by only simulating representative samples of the traces. In the area of embedded device development the uses cases of interest are known

from the beginning or may be combined from existing use cases. For example, a videoconference application may be composed of the already existing *audio phone call* use case and the *photo capturing* and *photo displaying* use cases.

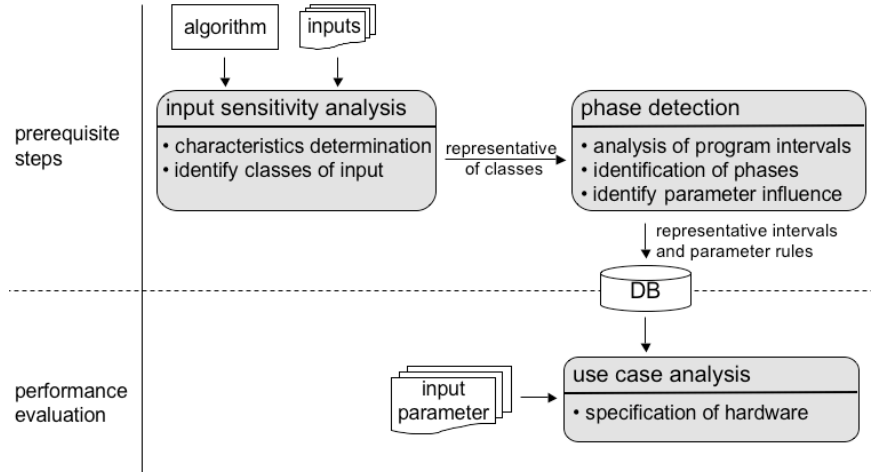


Figure 1: Overview of the main method steps and the final use case analysis.

In the following, the main features of the approach are described. Figure 1 shows an overview of the method. It can be divided into two parts. In the *prerequisite steps* a database is filled with representative intervals of program traces. In the final *performance evaluation* part performed by the developers in their daily work, the intervals stored in the database are used to predict the performance of different architectures with the help of trace-driven simulations. In the *input sensitivity analysis* step of the first part, architecture-independent characteristics of program executions for different input files are determined. Among other characteristics the *instruction mix* (e.g. arithmetic, logical, or memory instructions) as well as register access and usage characteristics like the *register definitions* and *register used* characteristics are calculated. The *register definitions* characteristics count how often a register was written and the *register used* characteristics state how often a register was used as an operand. These characteristics have been proposed and successfully applied in literature [LSC04]. The advantage of architecture-independent characteristics is that the influence of the executing architecture is reduced. In this way, the results can be transferred to other architectures, too. The result of this *characteristics determination* is a *characteristics vector* for each program-input pair. The dimension of these vectors is the number of calculated characteristics. Following the approach of [EVB03] a principal component analysis (PCA) is applied on the characteristics vectors. Since the characteristics vectors have more than 30 dimensions the principal component analysis is applied to visualise and interpret the data. The principal component analysis determines so called principal variables which are linear combinations of the original characteristic variables and eliminates the correlation between them. The dimensions of the characteristics vectors are reduced to the number of principal variables which cover an appropriate amount of the variance of the original characteristics variables. This

drastically reduces the dimensions of the vectors and enables the visualisation of the vectors in the *PCA space*. The visualisation of the characteristics vectors for the program-input pairs in the PCA space show the level of input sensitivity of the use cases. If the characteristics vectors for different inputs are similar and therefore close together, the input sensitivity of the use case is low. If the characteristics vectors differ and therefore scattered, the input sensitivity of the use case is high. In the latter case, representatives for each input class have to be selected. The input sensitivity of a use case can be mathematically represented by calculating the distance of each characteristics vector to the average characteristics vector of the use case.

The selected representatives of the input classes are the input for the second step in the prerequisite part, i.e. the *phase detection* step. In this step, characteristics of intervals of single program executions are analysed with the same techniques as in the input sensitivity step. The traces are split into intervals of instructions. An appropriate length of the intervals is gained by varying the number of instructions in the intervals until a clustering of the interval characteristics vectors is found. The analysis of the function call graph of the trace and the number of needed instructions of the functions provide a starting point for the interval length selection. For each interval the characteristics vector is calculated. Afterwards a principal component analysis is applied on these *interval characteristics vectors*. A cluster analysis of these vectors helps to identify the parameter dependent phases of the use case. The phases are weighted depending on the number of included intervals. In this way, it is possible to track the influence of changes in the input data e.g. the influence of smaller images or more audio frames. From these observations it is possible to derive rules that describe the influence of changes in the input data. For example, it may be observed that the weights of the parameter dependent phase for different audio inputs from the same input class depend linearly on the number of frames in the audio files. Representative trace intervals for the parameter dependent phases are stored in a database together with the rules that describe the influence of the parameters. These rules are used in the *performance evaluation* part of the method to predict the runtime or other performance metrics by just simulating the selected representative intervals of the phases and applying the projection rules. In this way, the developers do not need to provide real input data for the analysis of a use case, if they want to examine different inputs. Instead, they are enabled to state the important input data parameters only, e.g. image resolution or number of audio frames.

4 Evaluation

The suggested input sensitivity analysis is presented for four use cases, i.e. JPEG compression, JPEG decompression, GSM encoding, and GSM decoding. These use cases are the basic parts of a video conference application. The representative interval selection and the projection of performance metrics is exemplary shown for GSM encoding and JPEG compression. All analysed programs are part of the MiBench suite [GRE⁺01] which provides typical programs for different application areas of embedded and mobile devices. The JPEG algorithms are part of the consumer category of MiBench and the GSM algorithms are part of the telecommunication category. In case of the JPEG algorithms,

the analysed inputs are taken from the MiDataSet collection [FCOT07] which provides sets of input data for most of the MiBench programs. The parameters of the input images vary in several settings. Beneath the content and the size other parameter settings are varied, e.g. bits per pixel and the compression algorithm (progressive, baseline). In case of the GSM algorithms, no input data is provided by the MiDataSet collection therefore, we used audio samples from a spoken radio play. Beneath the content of the audio data and the number of audio frames, the number of audio channels is varied.

In the input sensitivity analysis step, architecture-independent characteristics are measured for a set of inputs for the use cases. Afterwards a principal component analysis is applied which reduces the dimensions of the characteristics vectors. Figure 2 shows the characteristics vectors for the four programs and several inputs in the PCA space. It can be seen that the use cases have different characteristics and input sensitivities. The characteristics vectors of the GSM encoding executions form two groups. The inner group input sensitivity is very low, because the characteristics vectors of each group are close together. The same observations hold for the GSM decoding algorithm. The characteristics of the executions of the JPEG decompression algorithm varies depending on the input data. The input sensitivity of the JPEG compression algorithm is lower than the input sensitivity of the decompression, but is higher than the inner group input sensitivity of the GSM algorithms.

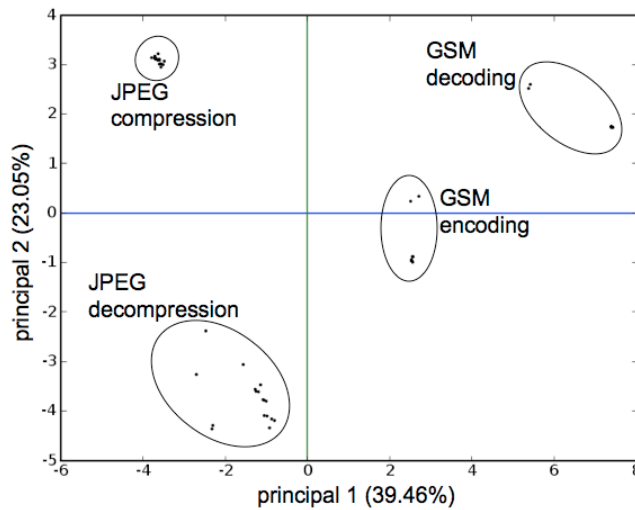


Figure 2: JPEG compression, JPEG decompression, and GSM encoding and decoding characteristics for several inputs.

In the phase detection step of the method, instruction intervals of representative program executions for each identified input class of an algorithm are analysed. In the following, the GSM encoding and the JPEG compression use cases are used to exemplarily show the existence of different input classes and the evaluation of the phase detection and the performance predictions.

4.1 GSM Encoding Use Case

This section shows the steps of the phase detection analysis for the GSM encoding use case and presents results of the projection using representative intervals compared to full simulations of the traces. Figure 3 presents a dendrogram of all analysed inputs for the GSM encoding use case. The x-axis states the different input files and the y-axis gives the distance between the detected clusters. The connections between the inputs represent the clustering for the iterative steps of the cluster algorithm. It can be seen that two input classes exist in the analysed audio files. The input parameter which leads to the differences in the characteristics is the number of audio channels. All inputs of the left cluster have two audio channels, whereas the input files of the right cluster have only one channel. By the calculation of the distances to the mean characteristics vector of the cluster for inputs with only one audio channel, one representative execution for the input class is selected.

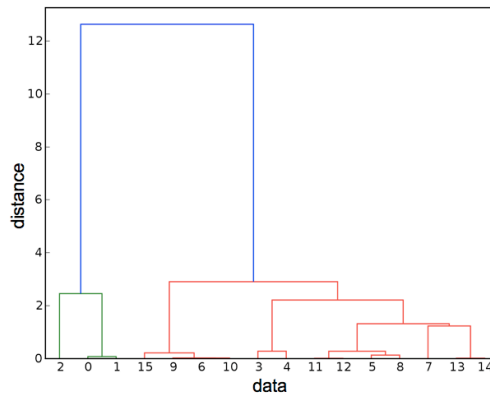


Figure 3: Dendrogram for the GSM encoding with different inputs.

For each instruction interval of the selected program execution, characteristics vectors are calculated and a principal component analysis is applied. In this way, intervals with similar characteristics are grouped close together in the PCA space. The length of the intervals is varied until an appropriate partition of the interval characteristics vectors is found. Thus, interval lengths which result in a strong clustering of the characteristics vectors have to be found. Figure 4 shows the interval characteristics vectors in the PCA space with an interval length of 200,000 instructions. It can be seen that there are no obvious groups of intervals. In contrast figure 5(a) shows the interval characteristics vectors with a length of 150,000 instructions for the same trace. For this interval length the characteristics vectors show a strong aggregation.

Figures 5(a) and 5(b) show the PCA space for two different input files of GSM. It can be seen that the interval characteristics grouping is very similar for both executions. The start and end phases have different characteristics and are clearly outside of the main phase. It can be observed that the number of intervals in the parameter dependent phase depends linearly on the number of frames in the input file. A representative interval is selected and stored in the database together with a rule describing the linear dependency on the number

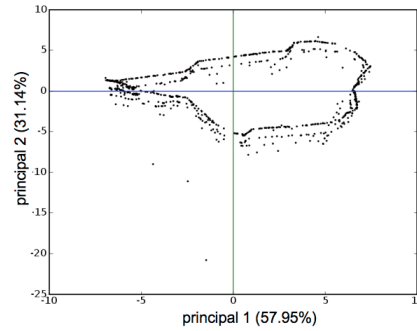


Figure 4: Interval characteristics vectors for an interval length of 200,000 instructions in the PCA space.

of frames. This information can now be used in the performance evaluation part of the method to predict performance metrics while just simulating representative intervals of the parameter dependent phase.

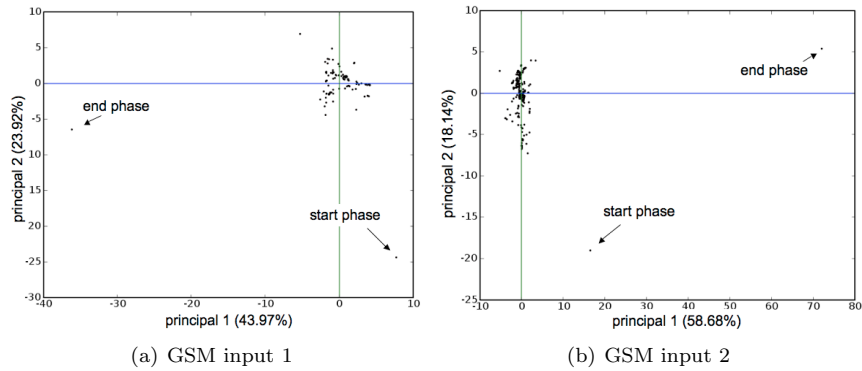


Figure 5: Interval characteristics vectors for an interval length of 150,000 instructions in the PCA space.

In the following, the error of the projection compared to simulated runtimes of the original traces are analysed for three different architecture configurations. Since ARM processors are widely deployed in mobile phones, different ARM9 architectures are used for the analysis. For example, the OMAP5912 board contains an ARM9 processor with a clock frequency of 192 MHz and an instruction cache of 16 kB as well as an 8 kB data cache. This configuration and two architectures with modified cache sizes are analysed. The representative sample of the parameter dependent phase is used for the prediction of the runtimes of other GSM inputs with different content and number of frames. By using the observed linear dependency on the number of frames and just simulating the representative interval from the parameter dependent phase, the runtimes are predicted with a mean relative error of less than 5% for all three architecture configurations. Table 1 presents the mean relative errors and the standard deviation for the analysis. By using the projection the number of

instructions to simulate is reduced to the length of the intervals and an appropriate amount of instructions to pre-fill the caches. Methods for the selection of an appropriate number of instructions to warm-up the caches are presented in the literature, e.g. [ELBJ05, WWFH03]. This results in a significant speedup of the performance evaluation, because the original GSM traces consist of millions of instructions. The simulation of the original traces last hours whereas the simulation and the projection of the representative intervals can be performed in seconds. Moreover, it is possible to predict the runtime on different architectures by just simulating the representative intervals and without the need to provide real input data. In case of the GSM example, the developers only has to set the number of frames of the input data.

instruction cache	data cache	mean rel. error	std.
16 kB	8 kB	3.82%	0.95%
16 kB	16 kB	4.04%	0.97%
8 kB	8 kB	1.10%	0.94%

Table 1: GSM encoding: mean relative errors and standard deviations of the projection using one representative interval compared to full simulations.

4.2 JPEG compression use case

The input sensitivity of the JPEG compression algorithm is not as low as for the inner class GSM encoding (cf. figure 2). Figure 6 shows a dendrogram of JPEG compressions for different input pictures. The diagram shows the existence of different input classes. In contrast to the GSM encoding use case, it is not possible to identify one single input parameter which classifies the input classes. The amount of bits per pixel and the type of compression algorithm (progressive or baseline) are candidates for a classification, but have to be analysed separately. The following analysis will focus on the large group of inputs on the right side of the dendrogram, i.e. input 0 - input 14. Nevertheless, predictions of the runtime of other input classes with a representative from the largest group are presented.

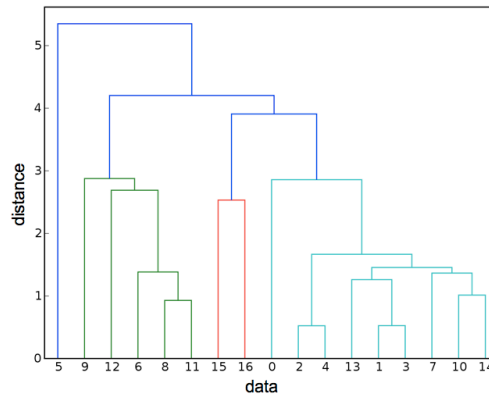


Figure 6: Dendrogram for the JPEG compression with different input pictures.

The characteristics vector of input 13 is the closest vector to the mean characteristics vector of the selected cluster. Therefore, input 13 is selected as representative execution and the interval characteristics vectors of this trace are analysed in the following phase detection step.

The number of instructions in an interval are again varied until an appropriate partition of the interval characteristics vectors is found. Interval lengths of 100.000 and 150.000 result in a clustering of the characteristics vectors. Figure 7(a) presents the interval characteristics vectors in the PCA space for the representative execution of the JPEG compression algorithm with an interval length of 150.000 instructions. It can be seen that there are two groups of characteristics vectors. Figure 7(b) shows the characteristics vectors in the PCA space for a trace of an input image with more pixels. The number of characteristics vectors in the two clusters increase linearly, so that a representative interval of the larger group is stored in the database together with a rule of the linear dependency on the number of pixels.

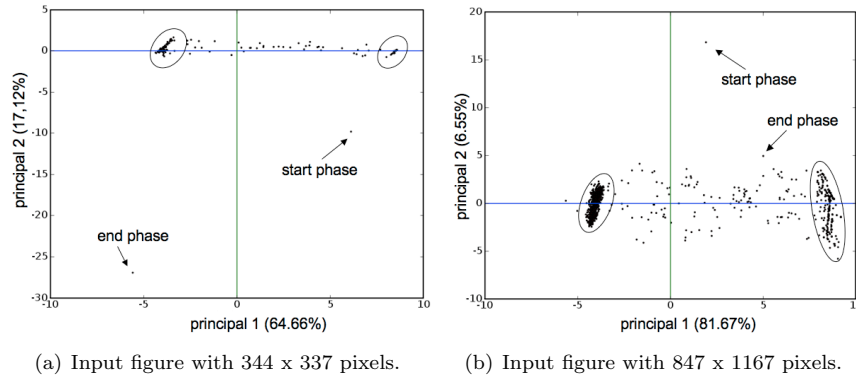


Figure 7: Interval characteristics vectors in the PCA space for two JPEG compressions.

In the following, the error of the projections compared to simulated runtimes for the JPEG compression use case is analysed. The projection of the runtime is not only compared with simulation results of the selected input class but also with simulation results of other input classes (cf. figure 6). The simulated and projected runtimes are analysed for three different configurations of an ARM9 processor (cf. section 4.1). Table 2 presents the mean relative errors and the standard deviations for these analyses. The runtime of executions for inputs of the same class compared to simulations of the corresponding trace are predicted with a mean relative error of less than 6% for all three architecture configurations. In case of runtime predictions for inputs from all input classes the mean relative error increases but is still less than 12%. Figures 7(a) and 7(b) show the existence of two parameter dependent clusters for an interval length of 150,000 instructions. By the selection of a representative interval per cluster and an appropriate weighting the prediction error may be decreased.

The input sensitivity analysis, the phase detection, and the performance evaluation steps have been evaluated for different algorithms and architecture configurations. The input sensitivity of four algorithms, i.e. JPEG compres-

cache		same input class		all inputs	
instr.	data	mean rel. error	std.	mean rel. error	std.
16 kB	8 kB	3.76%	2.48%	10.94%	8.36%
16 kB	16 kB	5.81%	3.47%	7.69%	3.96%
8 kB	8 kB	4.22%	2.99%	11.78%	8.83%

Table 2: Mean relative errors and standard deviations of the projection.

sion and decompression and GSM encoding and decoding, have been presented. JPEG compression and GSM encoding have been used to show the existence of different input classes of the use cases and to evaluate the phase detection step. Runtime predictions for three architectures with the help of selected representative instruction intervals have been compared to simulation results of the original traces.

5 Conclusion

The paper presented a method that eases and accelerates the performance analysis of embedded device use cases. Developers of embedded devices are enabled to analyse the performance of different architectures for use cases by just setting relevant input parameters. The presented approach base on the selection of representative samples of captured instruction traces of the use cases. This drastically reduces the amount of needed simulation time, so that the developers are enabled to perform extensive architecture explorations.

The method consists of an input sensitivity analysis which base on characteristics vectors of program executions. Principal component analyses are used to reduce the number of dimensions of the vectors to visualise executions with similar characteristics. The distances between the characteristics vectors show how strong the characteristics of the executions depend on the content of the input data. For use cases with a low input sensitivity, the same techniques are applied to intervals of a single program run to find representative phases and to determine the influence of changes in the input data. This knowledge is used to predict the performance without the need to provide real input data.

Future work will focus on the evaluation of more algorithms from the MiBench suite e.g. MP3 decoding and algorithms from the security category of MiBench. The choice of an appropriate size of the intervals for the analysis of single program runs will be automatised. Support of more than one representative interval for programs with more than one parameter dependent cluster is also an interesting topic.

References

- [ALE02] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An infrastructure for computer system modeling. *Computer*, 35(2):59–67, 2002.
- [Bel07] F. Bellard. Qemu homepage. <http://fabrice.bellard.free.fr/qemu/>, 2007.

- [DCD03] E. Duesterwald, C. Cascaval, and S. Dwarkadas. Characterizing and predicting program behavior and its variability. *Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques*, 2003.
- [ELBJ05] L. Eeckhout, Y. Luo, K. De Bosschere, and L. John. Blrl: Accurate and efficient warmup for sampled processor simulation. *Comput. J.*, 48(4):451–459, 2005.
- [EVB03] L. Eeckhout, H. Vandierendonck, and K. De Bosschere. Quantifying the impact of input data sets on program behavior and its applications. *Journal of Instruction-Level Parallelism*, 5:1–33, 2 2003.
- [FCOT07] G. Fursin, J. Cavazos, M. O’Boyle, and O. Temam. Midatasets: Creating the conditions for a more realistic evaluation of iterative optimization. In *International Conference on High Performance Embedded Architectures & Compilers*, January 2007.
- [GRE⁺01] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *WWC ’01: Proceedings of the Workload Characterization 2001*, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.
- [JE06] L. John and L. Eeckhout. *Performance Evaluation and Benchmarking*. CRC Press, Taylor and Francis Group, 2006.
- [LSC04] J. Lau, S. Schoemackers, and B. Calder. Structures for phase classification. In *Proceedings of the 2004 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 57–67, Washington, DC, USA, 2004. IEEE Computer Society.
- [PSM08] L. Pustina, S. Schwarzer, and P. Martini. A methodology for performance predictions of future arm systems modelled in uml. In *Proceedings of the 2nd Annual IEEE International Systems Conference Syscon 2008*, 2008.
- [SPC01] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *PACT ’01: Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques*, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.
- [SPHC02] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. *SIGOPS Oper. Syst. Rev.*, 36(5):45–57, 2002.
- [WWFH03] R. Wunderlich, T. Wenisch, B. Falsafi, and J. Hoe. Smarts: accelerating microarchitecture simulation via rigorous statistical sampling. *SIGARCH Comput. Archit. News*, 31(2):84–97, 2003.
- [YEL⁺06] J. Yi, L. Eeckhout, D. Lilja, B. Calder, L. John, and J. Smith. The future of simulation: A field of dreams. *Computer*, 39(11):22–29, 2006.

Fluid flow analysis of a model of a secure key distribution centre

Nigel Thomas* Yishi Zhao

Abstract

In this paper we consider the use of a fluid flow approximation based on ordinary differential equations (ODEs) derived from a model of a key distribution centre. The model is specified using the Markovian process algebra PEPA. The basic model suffers from the commonly encountered state space explosion problem when tackled using Markov chain analysis. Fluid flow analysis is therefore one possible mechanism for deriving approximate solutions for systems with large populations. The system is analysed numerically and results derived from solving the ODEs are compared with a queueing network approximation.

1 Introduction

In recent years a novel and intriguing approach to tackling the solution of a class of very large stochastic process algebra models has been developed based on the solution of ordinary differential equations [8]. This involves the analysis of the system as a deterministic fluid flow, rather than as a discrete stochastic system. Despite this apparent disparity between the analysis and the model, the results are often surprisingly good and allow approximate solution of systems which are not tractable by traditional means.

In this paper we aim to apply this form of analysis to a model of a key distribution centre, exploring the performance - security trade-off. It is clear that in order to add more functionality to a system that more execution time is required. However in the case of security, the benefit accrued from any additional overhead is not easy to quantify and so it is very hard for the performance engineer to argue that a particular performance target should take precedence over a security goal. Our initial inspiration for this work has been the study of the wide mouth frog protocol by Buchholz *et al* [2]. The authors used the stochastic process algebra PEPA to analyse timing properties of the protocol. Although their motivation was to investigate timing attacks, the models developed in [2] showed how authentication protocols can be modelled effectively in PEPA.

This paper is organised as follows. In the next section we introduce the system to be modelled, the key distribution centre (KDC). This is followed by a brief overview of the Markovian process algebra PEPA. Section 4 introduces the basic model of the KDC, followed by a simplified (equivalent) version and an approximation in Section 5. Some numerical results are presented in Section

*School of Computing Science, Newcastle University, UK. *Nigel.Thomas@ncl.ac.uk*

6, including comparison of the approximation results with simulation. Finally some conclusions are drawn and areas of further work described.

2 Key Distribution Centre

We now describe the specific problem we seek to model. This is the secure exchange of secret keys (also known as symmetric keys) using a trusted third party known as a key distribution centre (KDC). The protocol is illustrated below, following the description in [11].

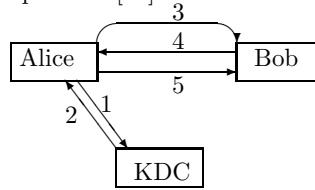


Figure 1: Key Distribution Scenario.

- Alice and KDC share a key K_A
 - Bob and KDC share a key K_B
1. Alice sends request to KDC with nonce N_1
 2. $E\{K_A\} [K_S | request | N_1 | E\{K_B\} [K_S | ID_A]]$
 - K_S is a session key for Alice and Bob to use.
 - Alice can't decrypt the part encode with Bob's key, she can only send it on.
 3. $E\{K_B\} [K_S | ID_A]$
 4. $E\{K_S\} [N_2]$
 5. $E\{K_S\} [f(N_2)]$

where,

- N_1 and N_2 are nonces (random items of data),
- ID_A is a unique identifier for Alice,
- $E\{K_A\}[X]$ denotes that the data X is encrypted using the key K_A , and
- $f(N_2)$ denotes a predefined function applied to the nonce N_2 , signifying that Alice has read the encrypted message sent by Bob.

The key features of this protocol are that only Alice can read the message sent by the KDC (2) as only Alice and the KDC know the key K_A . Included in this message is another message further encrypted with K_B , the key shared by Bob and the KDC. Alice cannot read this message, but instead forwards it to Bob (3). This message tells Bob that Alice is genuine (i.e. has communicated

with the KDC and displays a correct ID) and informs Bob of the session key; only Bob can read this message. Alice and Bob now both know the session key K_S and the remainder of the protocol ensures that Bob trusts Alice and the session key (and Alice trusts Bob).

3 PEPA

A formal presentation of PEPA is given in [7], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the type of the activity, α , and the rate of the associated negative exponential distribution, r . This rate may be any positive real number, or given as unspecified using the symbol \top .

The syntax for describing components is given as:

$$(\alpha, r).P \mid P + Q \mid P/L \mid P \bowtie_{\mathcal{L}} Q \mid A$$

The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P . The component $P + Q$ behaves either like P or like Q , the resultant behaviour being given by the first activity to complete.

The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ .

Concurrent components can be synchronised, $P \bowtie_{\mathcal{L}} Q$, such that activities in the cooperation set \mathcal{L} involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to activities of that type. $A \stackrel{\text{def}}{=} P$ gives the constant A the behaviour of the component P .

In this paper we consider only models which are cyclic, that is, every derivative of components P and Q are reachable in the model description $P \bowtie_{\mathcal{L}} Q$. Necessary conditions for a cyclic model may be defined on the component and model definitions without recourse to the entire state space of the model.

4 The model and its queueing network approximation

In [12] we developed three approaches to modelling multiple clients requesting session keys from the KDC. These approaches all model the same protocol and are notionally equivalent at the syntactic level (they have a form of *bisimilarity*). However, they are not isomorphic and hence can give different values for important performance metrics. The models specified in [12] suffer from the commonly encountered state space explosion problem. To counter this we have applied some simplification techniques to derive a form of the model which gives the same results to key steady state metrics [13]. This model is specified as follows:

$$KDC \stackrel{def}{=} (request, \top).KDC + (response, r_p).KDC$$

$$Alice \stackrel{def}{=} (request, r_q).(response, \top).Alice'$$

$$Alice' \stackrel{def}{=} (sendBob, r_B).(sendAlice, \top).(confirm, r_c).Alice''$$

$$Alice'' \stackrel{def}{=} (usekey, r_u).Alice$$

$$Bob \stackrel{def}{=} (sendBob, \top).(sendAlice, r_A).(confirm, \top).Bob'$$

$$Bob' \stackrel{def}{=} (usekey, \top).Bob$$

$$System \stackrel{def}{=} KDC \underset{\mathcal{L}}{\boxtimes} \left(Alice \underset{\mathcal{K}}{\boxtimes} Bob \parallel \dots \parallel Alice \underset{\mathcal{K}}{\boxtimes} Bob \right)$$

Where, $\mathcal{L} = \{request, response\}$, $\mathcal{K} = \{sendBob, sendAlice, confirm, usekey\}$.

Clearly the component *Bob* is almost redundant, and the sharing for the action *request* and its enabling in *KDC* has no effect on the behaviour of the model. Hence an even simpler equivalent specification would be:

$$KDC \stackrel{def}{=} (response, r_p).KDC$$

$$Alice \stackrel{def}{=} (request, r_q).(response, \top).Alice'$$

$$Alice' \stackrel{def}{=} (sendBob, r_B).(sendAlice, r_A).(confirm, r_c).Alice''$$

$$Alice'' \stackrel{def}{=} (usekey, r_u).Alice$$

$$System \stackrel{def}{=} KDC \underset{response}{\boxtimes} (Alice \parallel \dots \parallel Alice)$$

This model and the preceding one are clearly isomorphic, i.e. they have equivalent CTMCs with a one to one mapping between states and transitions. We can now apply the well known approximation technique of combining successive internal actions into a single action with a modified rate. This is equivalent to lumping states in the underlying Markov chain (Hillston [7] introduced the *weak isomorphism* equivalence for exactly this purpose). Thus we obtain the following simple form of the model.

$$KDC \stackrel{def}{=} (response, r_p).KDC$$

$$Alice \stackrel{def}{=} (response, \top).(\tau, r_x).Alice$$

$$System \stackrel{def}{=} KDC \underset{response}{\boxtimes} (Alice \parallel \dots \parallel Alice)$$

Where r_x is given by

$$r_x = \left(\frac{1}{r_q} + \frac{1}{r_B} + \frac{1}{r_A} + \frac{1}{r_c} + \frac{1}{r_u} \right)^{-1}$$

This model is equivalent to a simple closed queueing system with one queueing station (the KDC) and an exponential delay after service before returning to the queue. It is a simple matter to write down the balance equations for such a system.

$$(N - i)r_x\Pi_i = r_p\Pi_{i+1} \quad , \quad 0 \leq i < N$$

where Π_i is the steady state probability that there are exactly i jobs waiting for a response from the KDC and N is the number of pairs of clients (the number of instances of *Alice* in the above PEPA model specification). Thus it is possible to derive expressions for the average utilisation of the KDC and the average number of requests waiting for a response.

$$\Pi_0 = \left[N! \sum_{i=0}^N \frac{\rho^i}{(N-i)!} \right]^{-1}$$

and,

$$L = N!\Pi_0 \sum_{i=1}^N \frac{\rho^i i}{(N-i)!}$$

where $\rho = r_x/r_p$.

This approximation is, in fact, an $M/M/1/.N$ queue and the throughput and average response time are easily computed from the above expressions (see Mitrani [10] pages 195-197).

$$T = (N - L)r_x$$

and

$$W = \frac{N}{T} - \frac{1}{r_x}$$

We can easily increase the number of servers at the KDC in the PEPA specification.

$$System \stackrel{def}{=} (KDC || \dots || KDC) \underset{response}{\bowtie} (Alice || \dots || Alice)$$

In addition we must give the *response* action in *Alice* the rate r_p , rather than being passive.

$$Alice \stackrel{def}{=} (response, r_p).(\tau, r_x).Alice$$

This is because a passive action would be subject to the *apparent rate* in PEPA. Hence, K KDCs and 1 *Alice* would give rise to *response* occurring at rate Kr_p ; whereas if the rate is r_p in both *KDC* and *Alice*, then this problem does not arise.

Thus the approximation becomes an $M/M/K/.N$ queue, where K is the number of instances of the KDC component (i.e. servers at the KDC). Hence the balance equations become,

$$\begin{aligned} (N - i)r_x\Pi_i &= (i + 1)r_p\Pi_{i+1} \quad , \quad 0 \leq i < K \\ (N - i)r_x\Pi_i &= Kr_p\Pi_{i+1} \quad , \quad K \leq i < N \end{aligned}$$

Thus we can calculate Π_0

$$\Pi_0 = \left[N! \sum_{i=0}^{K-1} \frac{\rho^i}{(N-i)!i!} + N! \sum_{i=K}^N \frac{\rho^i}{(N-i)!K!K^{i-K}} \right]^{-1}$$

The average queue length can then be calculated by

$$L = N! \Pi_0 \left[\sum_{i=1}^{K-1} \frac{\rho^i i}{(N-i)!i!} + \sum_{i=K}^N \frac{\rho^i i}{(N-i)!K!K^{i-K}} \right]$$

The average response time and throughput can then be computed as before.

5 ODE analysis

Thus far we have considered a traditional approach to modelling and analysis. In this section we consider an alternative approach proposed by Hillston [8], based on the solution of ordinary differential equations (ODEs). In this style of model analysis, the model is expressed as a number of replicated components and the ODEs represent the flow between behaviours (PEPA derivatives) of the components. Thus, by solving the ODEs, it is possible to ‘count’ the number of components behaving as a given derivative at any given time, t . In the absence of oscillations, the limit, $t \rightarrow \infty$, then gives a steady state value.

It is important to make two crucial observations about this approach. Firstly, this is a fluid approximation, not discrete behaviour. Therefore, we observe a continuous evolution of a derivative, so we can, at any given time, see a fraction of an *Alice* behaving in some way, and another fraction behaving in another. Secondly the analysis is deterministic. Thus, not only will simulating such a system produce exactly the same results every time, but also if the rate of an action is r , then a component will have completely evolved (or *flowed*) into its derivative in exactly $1/r$ time units.

Rewriting our model, removing redundancy and naming each derivative of *Alice* (for clarity) we get:

$$\begin{aligned} KDC &\stackrel{def}{=} (response, r_p).KDC \\ Alice &\stackrel{def}{=} (request, r_q).Alice_1 \\ Alice_1 &\stackrel{def}{=} (response, r_p).Alice_2 \\ Alice_2 &\stackrel{def}{=} (sendBob, r_B).Alice_3 \\ Alice_3 &\stackrel{def}{=} (sendAlice, r_A).Alice_4 \\ Alice_4 &\stackrel{def}{=} (confirm, r_c).Alice_5 \\ Alice_5 &\stackrel{def}{=} (usekey, r_u).Alice \end{aligned}$$

The system is then defined as:

$$KDC[K] \underset{response}{\boxtimes} Alice[N]$$

Where, K is the number of *KDC*s (hitherto $K = 1$) and N is the number of client pairs (*Alices*’s). It is then a simple matter to write down the ODEs for

this system as follows.

$$\begin{aligned}
 \frac{d}{dt} Alice &= r_u Alice_5(t) - r_q Alice(t) \\
 \frac{d}{dt} Alice_1 &= r_q Alice(t) - r_p \min(KDC(t), Alice_1(t)) \\
 \frac{d}{dt} Alice_2 &= r_p \min(KDC(t), Alice_1(t)) - r_B Alice_2(t) \\
 \frac{d}{dt} Alice_3 &= r_B Alice_2(t) - r_A Alice_3(t) \\
 \frac{d}{dt} Alice_4 &= r_A Alice_3(t) - r_c Alice_4(t) \\
 \frac{d}{dt} Alice_5 &= r_c Alice_4(t) - r_u Alice_5(t) \\
 \frac{d}{dt} KDC &= 0
 \end{aligned}$$

There are a number of approaches to solving this set of ODEs. For simplicity we have simulated over a suitably long time frame until we observe the long run (steady state) behaviour. In doing so we need to be careful that in discretizing time we make the time step sufficiently small so as to not alter the system behaviour. Typically we take the time step, δt , such that, $\delta t \leq 1/(r_{max}N)$, where $r_{max} = \max(r_q, r_p, r_B, r_A, r_c, r_u)$.

In our analysis we are interested primarily in the number of client pairs awaiting a response from the *KDC* (or *KDC*'s). This is represented in the model by the number of *Alice*₁'s; $L(N) = Alice_1(t \rightarrow \infty)$ when there are N client pairs (*Alice*'s) in the population. From this we can derive the average response time which can be compared with that derived from the queueing network approximation. We compute the average response time for a system of N client pairs and one *KDC* server ($K = 1$), $W(N)$, as follows;

$$W(N) = \frac{L(N-1) + 1}{r_p}$$

This computation is based on the queueing theory result of an arrival as random observer, see Mittrani [10] page 141 for example. For $K > 1$ the computation is only slightly more complex. If the random observer sees a free server, then the average response time will be the average service time. However, if the random observer sees all the servers busy, then the average response time will be the average service time plus the time it takes for one server to become available (including scheduling the other jobs waiting ahead of the random observer).

$$\begin{aligned}
 W(N) &= \frac{1}{r_p}, \quad L(N-1) + 1 \leq K \\
 W(N) &= \frac{1}{r_p} + \frac{L(N-1) + 1 - K}{Kr_p} = \frac{L(N-1) + 1}{Kr_p}, \quad L(N-1) + 1 > K
 \end{aligned}$$

It is a feature of the fluid flow approximation that (for $t > 0$) the *KDC* will never be idle, but instead will always have some fluid flowing through it. As such we are unable to compute the utilisation of the *KDC* directly. This is clearly a limitation of this form of analysis.

6 Numerical results

Figure 2 shows the evolution over time of the number of clients awaiting a response as derived from the ODE analysis. Initially all the clients are behaving as *Alice*, hence $Alice_1(0) = 0$. Shortly after the start there is a large influx of fluid into *Alice*₁ before the system settles into a stable flow. Interestingly this initial surge is much more pronounced when $r_p = 4$ than $r_p = 1$. This is clearly due to the fact that the flow out of *Alice*₁ is much greater when $r_p = 4$.

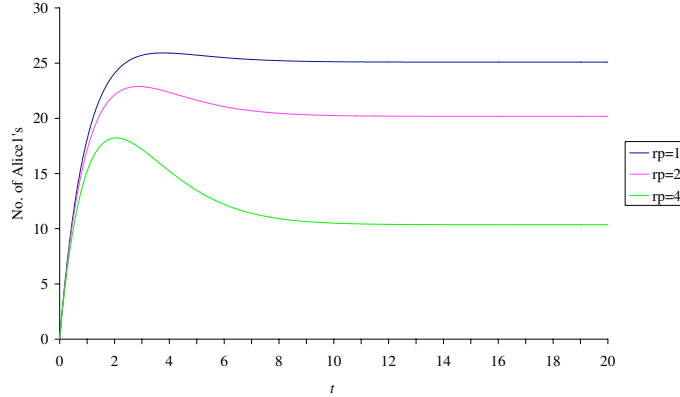


Figure 2: Number of waiting clients over time, $N = 30$, $r_q = r_B = r_A = r_c = 1$ and $r_u = 1.1$

Figure 3 shows the average response time calculated by the ODE method, compared with the queueing approximation described earlier. This approximation has previously been compared with simulation and shown to be accurate to within the 95% confidence interval of the simulation [13].

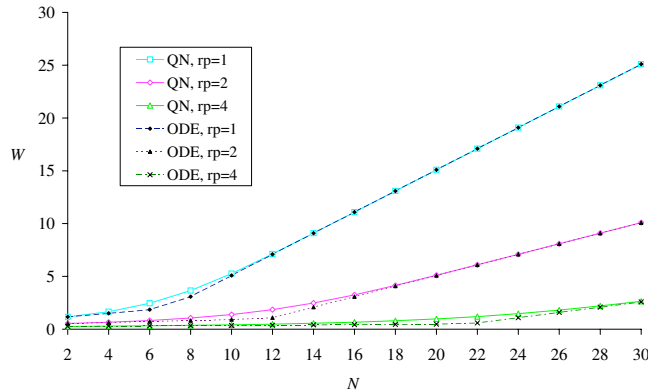


Figure 3: Average response time calculated by the ODE method and QN approximation, $r_q = r_B = r_A = r_c = 1$ and $r_u = 1.1$

We expect the ODE method to be accurate when N is large. Figure 3 shows that it is possible to generate accurate results even when N is quite small. However, there is a clear difference between the two methods where the gradient changes. This is shown more explicitly in Figure 4, where the evolution of the ODEs is compared with the stochastic simulation of the PEPA model [1]

derived directly using the PEPA Eclipse Plug-in. When N is sufficiently far from the gradient change there is good agreement between the ODE solution and the stochastic simulation. However, at $N = 6$ the divergence is significant; the stochastic simulation never achieves the lower queue length predicted by the ODEs.

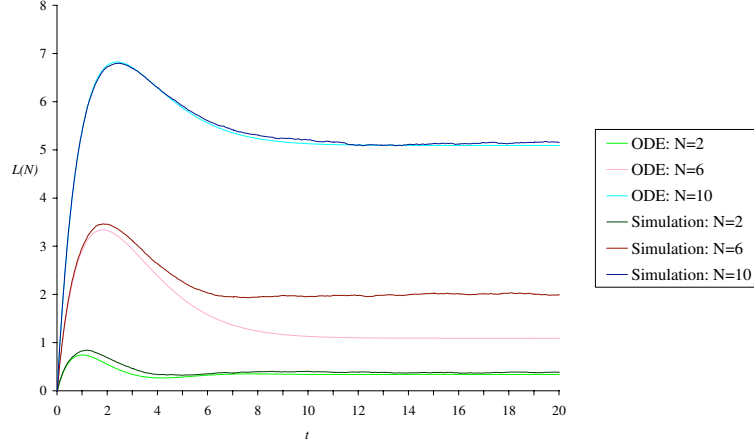


Figure 4: Number of waiting clients over time, $r_p = r_q = r_B = r_A = r_c = 1$ and $r_u = 1.1$

It is of clear practical importance to be able to predict the divergence. This point, N^* , can be estimated using the method of asymptotic bounds of closed queueing networks (see Haverkort [6] pages 245-246 for example).

$$N^* = K + \frac{Kr_p}{r_x} = K + Kr_p \left(\frac{1}{r_q} + \frac{1}{r_B} + \frac{1}{r_A} + \frac{1}{r_c} + \frac{1}{r_u} \right)$$

Below N^* the asymptotic bound is given as

$$L(N) = \frac{Nr_x}{r_x + r_p}$$

Above N^* the asymptotic bound is given as

$$L(N) = \frac{Nr_x - Kr_p}{r_x}$$

These bounds can also easily be found by solving the ODEs analytically in the limit $t \rightarrow \infty$, where the $\min(KDC(t), Alice_1(t))$ term is replaced with $Alice_1(t)$ and $KDC(t)$ respectively. Thus, in this instance at least, the ODE solution is giving an alternative means for calculating known asymptotic results for closed queueing networks. Note that $W(N)$ is computed from $L(N + 1)$, and so in Figure 3, the divergence occurs at approximately 6.91 ($r_p = 1$), 11.82 ($r_p = 2$) and 21.64 ($r_p = 4$), i.e. $N^* + 1$.

We have also compared the two methods for larger values of N and have found there to be almost no difference for $N > 40$ for the parameters used here. It is important to note that there are numerical issues with computing the queueing approximation due to the difficulty of handling large factorials

(and their reciprocals) and these problems do not occur with the ODE solutions or asymptotic results. Thus, as long as we avoid the region around N^* , the ODE solution is giving accurate results without problems with scalability.

6.1 Multiple KDC servers

We now turn our attention to the consideration of multiple servers at the KDC. In particular we would wish to know if it is more beneficial to increase the number of servers or increase the speed of the server. It is well known that for an M/M/K queue, it is preferable to have 1 server serving at rate μ than K servers serving at rate μ/K . This is because if there are less than K jobs in the queue then some of the K servers will be idle, thus reducing the overall service rate. In the ODEs above this is evident in $r_p \min(KDC(t), Alice_1(t))$. If $Alice_1(t) > K$ then all K servers are in use and the flow rate from the KDC would be Kr_p . However, if $Alice_1(t) < K$ then fewer servers would be in use and the rate would be $r_p Alice_1(t)$.

Figure 5 shows the proportion of Alices waiting at the KDC (i.e. $L(N)/N$) for $K = 1$ with $r_p = 4$ and $K = 4$ with $r_p = 1$ for both the queue approximation and the ODE solution. When N is large (in this case $N \geq 25$) the ODE values are the same, however for smaller N the single faster server is seen to perform better (for the reason discussed above). The reason the ODE values are identical for large N is simply that the fluid level of Alices waiting at the KDC will never fall below K in the ODE solution. The values for the QN approximation differ slightly from each other, even when $N = 40$. This is because even at this load there is still the chance that the queue will fall below 4 requests for short periods. Clearly, as N increases the probability that this happens will become increasingly insignificant and hence the values will converge.

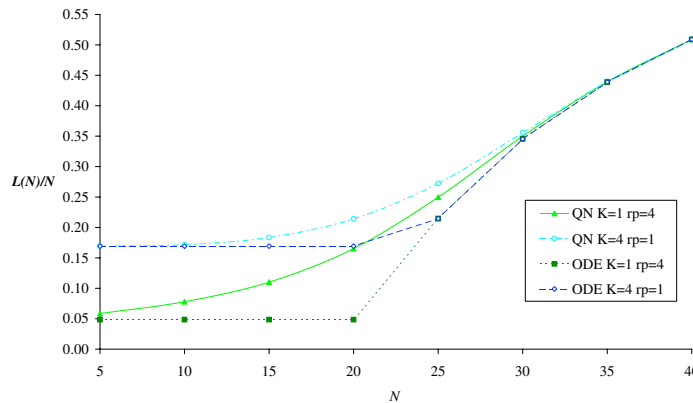


Figure 5: Proportion of $Alice_1$ components, calculated by the ODE method and QN approximation, $r_q = r_B = r_A = r_c = 1$ and $r_u = 1.1$

There is a clear divergence between the ODE and QN results around the change in gradient as we have already observed in Figure 3. Figure 6 shows this in more detail for the average queue size. Note that although the two ODE solutions converge at $N = 25$, there is still a significant difference with the QN solutions at this point, near to N^* .

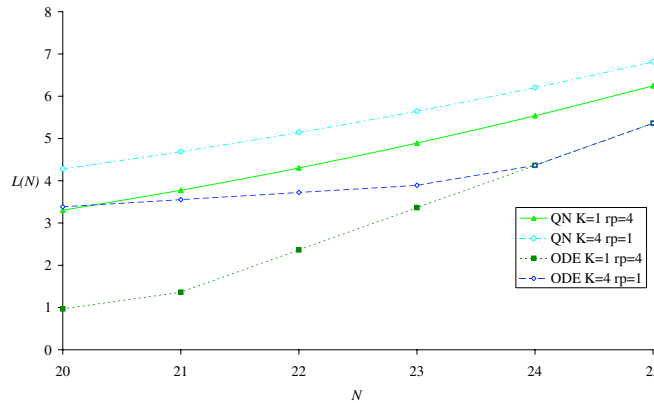


Figure 6: Average queue length calculated by the ODE method and QN approximation, $r_q = r_B = r_A = r_c = 1$ and $r_u = 1.1$

7 Conclusion and further work

In this paper we have shown how a model of a key distribution centre can be solved approximately using ordinary differential equations. We have compared the results with those derived from a scalable queueing approximation. The ODE approach has two main limitations. Firstly, it is not always as accurate as the queueing approximation and secondly, we have not been able to obtain all our desired metrics. However, the ODE approach does not suffer the same numerical problems as the queueing approximation, is extremely efficient to solve and is shown to be extremely accurate when the number of clients is large. By using the asymptotic results, it is possible to compute the metrics of interest extremely efficiently.

The next step in the study of this model of key distribution is to explore the use of a cost model to better understand the relationship between the KDC server capacity and the quality of service. To do this we will introduce costs for waiting jobs and providing service capacity (e.g. the number of servers) and minimise the resulting function to produce an optimal service capacity for a given cost structure. In doing this, the asymptotic results derived from the ODE solution are likely to provide the most tractable approach. We also aim to extend this analysis to a more general form of secure protocol in the area of non-repudiation. A parallel line of investigation involves exploring more general closed queueing network models in PEPA to discover if the ODE approach facilitates an efficient means of finding asymptotic results as has been demonstrated in this paper. This would be a worthwhile study as in process algebra, it is not always clear that a model can be represented as a queueing network and so such a result could potentially open an alternative means of analysis for this class of model.

Acknowledgements

The authors are indebted to A. Clark, A. Duguid, S. Gilmore and M. Tribastone of the University of Edinburgh for invaluable comments.

References

- [1] J. Bradley, S. Gilmore and N. Thomas, Performance analysis of Stochastic Process Algebra models using Stochastic Simulation, in: *Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium*, IEEE Computer Society, 2006.
- [2] M. Buchholtz, S. Gilmore, J. Hillston and F. Nielson, Securing statically-verified communications protocols against timing attacks, *Electronic Notes in Theoretical Computer Science*, 128(4), Elsevier, 2005.
- [3] G. Clark and S. Gilmore and J. Hillston and N. Thomas, *Experiences with the PEPA Performance Modelling Tools*, IEE Proceedings - Software, pp. 11-19, 146(1), 1999.
- [4] S. Dick and N. Thomas, *Performance analysis of PGP*, in: F. Ball (ed.) *Proceedings of 22nd UK Performance Engineering Workshop*, Bournemouth University, 2006.
- [5] W. Freeman and E. Miller, An Experimental Analysis of Cryptographic Overhead in Performance-critical Systems, *Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE Computer Society, 1999.
- [6] B. Haverkort, *Performance of Computer Communication Systems: A model Based Approach*, Wiley, 1998.
- [7] J. Hillston, *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.
- [8] J. Hillston, Fluid flow approximation of PEPA models, in: *Proceedings of QEST'05*, pp. 33-43, IEEE Computer Society, 2005.
- [9] C. Lamprecht, A. van Moorsel, P. Tomlinson and N. Thomas, Investigating the efficiency of cryptographic algorithms in online transactions, *International Journal of Simulation: Systems, Science & Technology*, 7(2), pp 63-75, 2006.
- [10] I. Mittrani, *Probabilistic Modelling*, Cambridge University Press, 1998.
- [11] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, 1999.
- [12] Y. Zhao and N. Thomas, Modelling secure secret key exchange using stochastic process algebra, in: E. Pereira and R. Pereira (eds.) *Proceedings of 23rd UK Performance Engineering Workshop*, Edge Hill University, 2007.
- [13] Y. Zhao and N. Thomas, Approximate solution of a PEPA model of a key distribution centre, in: *Performance Evaluation - Metrics, Models and Benchmarks: SPEC International Performance Evaluation Workshop*, pp. 44-57, LNCS 5119, Springer-Verlag, 2008.

Queuing Networks for the Performance Evaluation of Database Designs

Rasha Osman* Irfan Awan† Michael E. Woodward‡

Abstract

This paper proposes a novel approach to evaluate the performance of database designs using queuing networks. It differs from other methods of database system performance evaluation in that the performance assessment is specifically targeted at the database design, not at the database system software architecture. The value of the proposed method for the performance evaluation of database designs is demonstrated through comparisons with the implementations of the TPC-C benchmark.

1 Introduction

Research in the area of database (DB) system performance evaluation has been limited, primarily by the focus on database management system (DBMS) component performance in the database field [1] and the concentration of the performance engineering community on software system architectures and design models [2]. Performance evaluation of *database designs*, as described in this paper, to the best of our knowledge, has not been previously addressed.

Conventional software testing methods do not aid in database design assessment as software testing focuses on the functionality of the system [3] and how well it conforms to its requirements. Performance requirements are tested through software performance engineering techniques [4], but these techniques, again, focus on the functionality of the whole system. No efforts are made towards the changes that occur inside the database and their effect on overall system performance, as demonstrated in [4] and [2].

Moreover, most of the research work in database system performance evaluation [5-10] assesses the performance of these systems at the software architecture level and is mostly concerned with physical capacity planning, not database design troubleshooting [11]. Contrary to this trend, this paper proposes an original method to evaluate the performance of database designs in early development phases using queuing networks.

The intention of this paper is to establish the validity of our method in effectively modelling database design performance as a first step in achieving a complete framework for the performance evaluation of database designs and database systems. This work is an implementation of the ideas described in [11]. The contributions of this paper are as follows:

- We demonstrate the suitability of queuing networks for database design performance modelling.

* School of Informatics, University of Bradford, r.i.m.osman@bradford.ac.uk

† School of Informatics, University of Bradford, i.u.awan@bradford.ac.uk

‡ School of Informatics, University of Bradford, m.e.woodward@bradford.ac.uk

- We propose a novel performance evaluation model of database designs using queuing networks.
- We present an empirical evaluation of our technique by applying it to the Transaction Processing Performance Council (TPC) TPC-C database system benchmark, demonstrating the merit of our approach, as well as presenting performance measures for the TPC-C benchmark queuing network model.

The rest of this paper is organized as follows: in Section 2, the modelling technique is described. Experimental results are in Section 3. Conclusions and future work are in Section 4.

2 The Proposed Approach

Database system performance is usually measured in terms of query and transaction response time; the major indicator of a system capacity problem. After a database system exhibits a performance problem, the main effort of post-deployment performance tuning is concentrated on the revision of the design of the database and the transactions running against the database [12-15]. Hence, if the flaws of the database design were discovered before system implementation and deployment, some of the post-deployment performance problems would have been avoided. The database design artifacts are the main contributors to performance problems; therefore, an early evaluation of their performance coupled with the knowledge of the application design is a major factor in the reduction of post-deployment database tuning.

Now, consider a database design with tables and transactions that access these tables. A valid assumption would be:

Total response time of a transaction =

$$\sum_{i=1}^n \left(\begin{array}{l} \text{total time to wait for access to table}_i + \\ \text{total time to access the data of table}_i + \\ \text{total time to return data to the client from table}_i \end{array} \right) + \text{total time to process procedural statements on the client}$$

where table 1,2, ..., n are the tables accessed by the transaction.

But:

- total time to wait for access to the table = total time waiting for other transactions to complete their access to the table; this can be considered as the *queuing time*;
- total time to access the data = total number of disk accesses (I/O DB pages) to fetch the data into main memory \times the duration of one disk access + total time to complete the operations on the data; this can be considered as the *service demand*;
- total time to return data to the client depends on the rate the data oscillates between the client and the server;
- total time to process procedural statements depends on the processing speed of the client.

Therefore, the relationship between tables and transactions can be represented as a queuing system and modelled using a queuing network. In the queuing network the tables will represent the shared resources, i.e. the servers, and the transactions that use these resources are the customers. Total time to return data to the client and process

procedural statements can be aggregated for each transaction as the client think time or added as a delay resource in the queuing network.

Performance modelling of database designs is possible because transaction service demands on their relevant tables can be estimated from the procedural structure of the transaction design, i.e., from the SQL statements, the procedural statements and the structure and relationships between tables by using database query optimization techniques [13, 14, 16]. Disk I/O cost is the dominant factor [16, 17] in query execution costs, especially for large databases; this is the cost criteria that is used to calculate service demands for transactions on the relevant tables for our queuing network models. Other performance evaluation inputs, e.g., frequencies and counts of transactions, number of transaction invocations, user population, etc, are available or can be calculated from the application design [4]. In our approach, we limit ourselves to the query optimization techniques that are available to database designers in commercial DBMS, see [13], [14], or [16].

2.1 Building the Queuing Network

The steps to build a queuing network model for a database design are described next.

The Input

A database design composed of:

- Tables:
 - Structure, data types, attribute selectivity, etc.
 - Expected number of rows and record length.
 - Index types and structure.
- Transactions:
 - Rate of occurrence or % of total transactions.
 - Structure :
 - SQL statements:
 - Tables accessed.
 - Join/retained attributes: sequence, selectivity.
 - Access path: can be calculated in I/O DB pages.
 - Procedural statements.

The Method

The servers in the queuing network model are classical M/M/1 queues with parameters specified based on the database design.

1. Specify server parameters:
 - Servers: each table in the database design is a server in the queuing network; partitioned or replicated tables are represented as separate servers.
 - Customer classes: each transaction type is considered as a different customer class: transaction types that access identical tables with equal service demands may be considered as one class.
 - Queuing discipline is FCFS: DBMS use queues to control access to data objects; a new transaction is given access to a data object depending on the state of the current transactions waiting to access or currently accessing the data object. Depending on the concurrency control mechanism implemented by the DBMS, access is either granted immediately to the new transaction or it is forced to queue behind the

current transactions [13]. FCFS abstracts this in forcing all transactions to wait.

- Queue length is infinite: this is based on the assumption that aborts due to deadlocks are rare in DBMS [13] and that system overload causes long response times instead of transaction aborts.
2. Specify performance characteristics for the customer classes:
 - Transaction service demands on each server: the total cost of executing the SQL statements in terms of I/O DB pages (service demands are assumed to be exponentially distributed with the mean being the calculated I/O DB page cost \times the duration of one disk page access).
 - Transaction arrival rates (open queuing network) or number in system (closed queuing network).
 - Transaction think times.
 3. Specify the routing table for the customer classes; i.e. the order in which the transactions access their tables.
 4. Solve the queuing network model.

The Output

Depending on the complexity of the queuing network model and the solution method used, some possible outputs are:

- For each table
 - Bottleneck resource
 - Total access compared to other tables
 - Mean queue length
 - Mean waiting time to access the table
- For each transaction
 - Mean response time
 - Mean waiting time to access each table
 - Response time distribution

3 Validation of the Approach

The Transaction Processing Performance Council (TPC) TPC-C benchmark revision 5.8.0 [18] is used as an example of a database system design. The TPC-C disclosed implementations in [19-23] are used as examples of the implemented database system. The TPC-C benchmark is a design specification of an order-entry system. The specification is composed of [18]:

- 9 tables (WAREHOUSE, DISTRICT, CUSTOMER, HISTORY, ORDER, NEW-ORDER, ORDER-LINE, STOCK, ITEM);
- 5 transactions (New-Order, Payment, Order-Status, Delivery, Stock-Level).

A brief description of the transactions is in Table 1. The details of the design of the tables, the relationships between them, and the details of transaction functionality can be found in [18].

The TPC-C benchmark also includes performance specifications related to the implementation of the database system such as keying and think time distributions for transactions and the probability of operations on the database and the probability of choosing the values of the parameters for the transactions [18].

Table 1. Summary of the TPC-C benchmark transactions.

Transaction	Description	Min. % of the total number of transactions
New-Order	Initiates a new order.	No minimum
Payment	Updates the customer's balance and reflects the payment on the district and warehouse sales statistics.	43
Order-Status	Queries the status of a customer's last order.	4
Delivery ¹	Processes a batch of 10 new orders, one for each district for a given warehouse.	4
Stock-Level	Counts the number of items in the last 20 orders in a district that fall below the stock threshold.	4

3.1 Experimental Results

Using the transaction descriptions of the TPC-C benchmark in [18] and the order of execution, index structures and SQL statements described in [19-23] in addition to the assumptions detailed in the appendix, the service demands (number of I/O DB pages) are calculated for the transactions on each table by using query optimization techniques. Details of applying query optimization techniques to estimate query DB page cost can be found in [13, 14, 16]. The cost model is described in [13].

Applying the steps described in Section 2 we get the service demands for all the transactions (Table 2) and the multi-class queuing network of Fig. 1. The TPC-C benchmark specifies that the transaction think times follow an exponential distribution and we have assumed that the service demands for the transactions on the tables are exponentially distributed with means as calculated in Table 2.

Table 2. Service demands for the TPC-C benchmarks.

Transactions	Service Demands (in DB pages) ²								
	I	II	III	IV	V	VI	VII	VIII	IX
New-Order	1.2	2.2	1.2	0	2.2	2.2	7.3	7.3	4
Payment	2.2	2.2	12.76	2	0	0	0	0	0
Order-Status	0	0	15.51	0	3.45	0	4.63	0	0
Delivery	0	0	7.3	0	13.2	19.7	15.4	0	0
Stock-Level	0	1.2	0	0	0	0	10.63	200.76	0

I = WAREHOUSE, II = DISTRICT, III = CUSTOMER, IV = HISTORY, V = ORDER, VI = NEW-ORDER, VII = ORDER-LINE, VIII = STOCK, IX = ITEM

Figure 1 is a multi-class queuing network, which was solved using simulation. The queuing network was simulated with 9 servers (tables), 5 classes (transactions) and 1500 users (terminals) using QNAP2, a discrete-event simulator for queuing networks [24], at 95% confidence interval. The small number of users (terminals) is due to the limitations of the QNAP2 package. The implementation in [19] has 80,000 users (terminals).

The TPC-C benchmark only specifies the disclosure of response times for each transaction, which is what the simulation measured. Figures 2a – 2e show the response time frequency distribution for the QNAP2 simulation and the disclosed

¹ The Delivery transaction is a deferred execution transaction with its response time calculated when it is queued for service (deferred response time) and when it finishes service (interactive response time).

² These values are multiplied by the duration of one disk page access to give the final service demands used in the simulation (see the appendix).

results in [19] for the transaction classes. In Fig. 2a - 2e and Fig. 3b, we have used HP1 to reference [19].

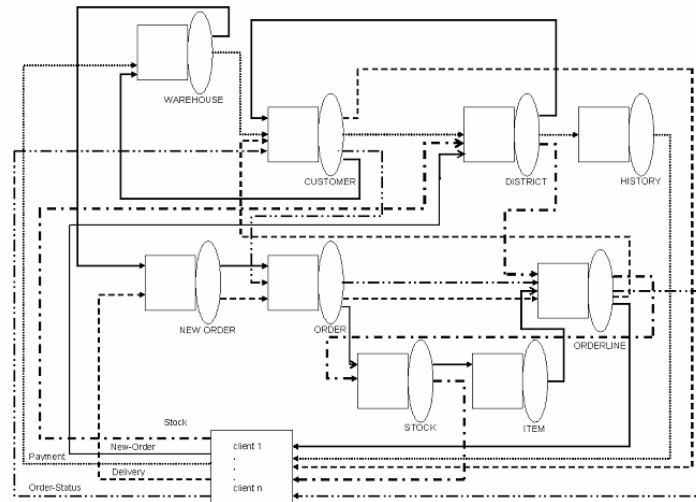


Fig. 1. Multi-class queuing network model for the TPC-C benchmark.

Due to the fact that actual data is not available for the TPC-C disclosed implementations, the comparison between our simulated model and that of [19] is conducted by using the graphs of these disclosed reports. The TPC-C specification states: for the response time frequency distribution graphs, the x-axis represents the transaction response time and must range from zero to four times the measured 90th percentile of the measured response times for that transaction and that at least twenty different intervals, of equal length, are to be reported [18]. The simulation graphs were plotted in this way, therefore a comparison between the different response time frequency distributions is possible.

From Fig. 2a - 2e, the response time frequency distributions for the simulated queuing network model of the database design are similar to those disclosed in [19]. This is in spite of the fact that the scales on the graphs are different. In addition, the implementations in [20-23] exhibit the same performance behavior as that of [19]. Hence, the simulated queuing network model of the database design exhibits the same performance behavior as that of the actual implemented database systems in [19-23].

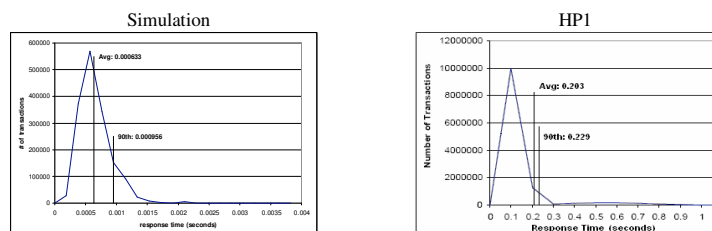


Fig. 2a. The response time frequency distribution for the New-Order transaction.

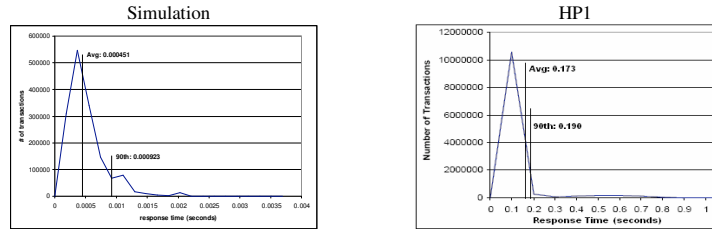


Fig. 2b. The response time frequency distribution for the Payment transaction.

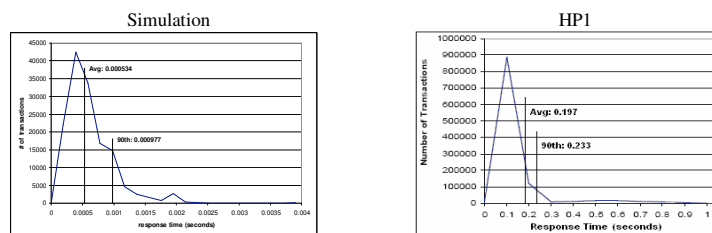


Fig. 2c. The response time frequency distribution for the Order-Status transaction.

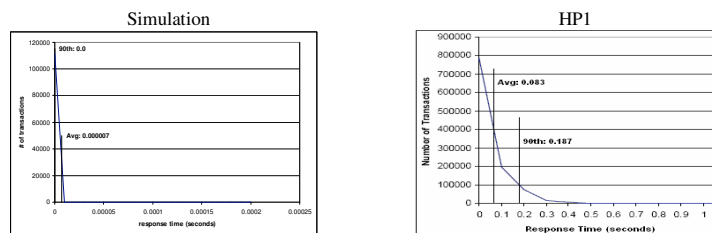


Fig. 2d. The response time frequency distribution for the Delivery (deferred) transaction.

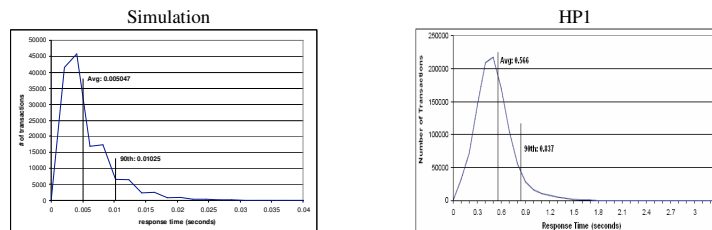


Fig. 2e. The response time frequency distribution for the Stock-Level transaction.

Figure 3a shows the average response time for the transactions of the simulated queuing network model of the TPC-C benchmark. Figure 3b, depicts the average response times for the TPC-C benchmark transactions disclosed in [19-23] (respectively HP1, HP2, IBM1, IBM2 and Bull) as well as the overall average of these disclosed results. These disclosed results utilize the TPC-C benchmark implementation for the Oracle DBMS. From Fig. 3b, the average response times of the transactions have a certain pattern in relation to each other in all implementations. The New-Order, Payment and Order-Status transactions have very similar average

response times, while the Delivery (deferred) transaction has the least average response time, the Stock-Level transaction has, on average, the longest average response time and the Delivery (interactive) falls between the Delivery (deferred) and the New-Order, Payment and Order-Status transactions.

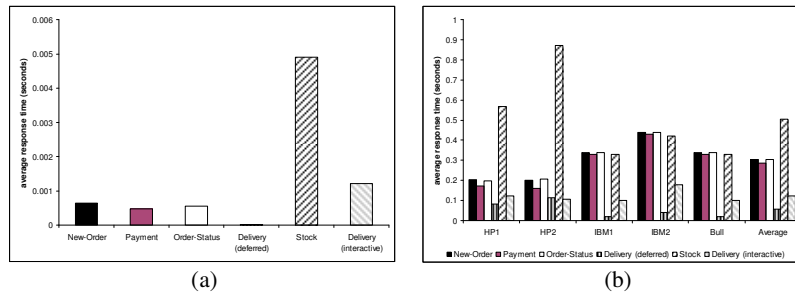


Fig. 3. (a) Simulation results for the average response time for the TPC-C benchmark transactions. (b) Average response time for the TPC-C benchmark transactions in [19-23].

From Fig. 3a the average response times given by the simulated queuing network model of the TPC-C benchmark design have this same pattern. There is a difference in the Stock-Level and the Delivery (interactive) transactions, which show relatively long average response times compared to the pattern shown in the implementations. This is not surprising, since the Stock transaction is a heavy JOIN transaction; it queries the DISTRICT, ORDER-LINE, and STOCK tables, with its largest service demand on the STOCK table (see Table 2). This difference is due to the fact that the implementations in [19-23] keep the STOCK table in the DBMS buffer at all times, therefore the Stock-Level transaction is faster. Buffering is not considered in the model at this stage, hence, the Stock-Level transaction runs faster in the implementations than in the model. This result was to be expected.

The Delivery (interactive) transaction is implemented in [19-23] using the Oracle PL/SQL FORALL construct which improves performance due to the elimination of context switches that usually occur in the execution of SQL statements in PL/SQL FOR LOOPS. There is no official measure of the expected performance gain that is achieved when implementing the TPC-C benchmark using the Oracle PL/SQL FORALL construct instead of the Oracle PL/SQL FOR LOOP (no TPC-C disclosed reports are available for older versions of the Oracle DBMS). Therefore, we have assumed a 66% performance gain when using the Oracle PL/SQL FORALL construct over the use of the Oracle PL/SQL FOR LOOP, based on the average performance gain described in the Oracle DBMS literature [25-27]. This gave the results in Fig 3a.

From the previous results, the queuing network model of the TPC-C benchmark database design exhibits comparable performance behavior and a similar transaction average response time pattern. The queuing network model was able to capture the expected behavior of the database transactions, using the details of the database design, the transaction DB I/O page costs, and the assumptions detailed in the appendix without considering the level of detail of the TPC-C benchmark disclosed implementations. This demonstrates the ability of the model to represent the database system.

3.2 Performance Measures

The simulation of the TPC-C benchmark queuing network was run for 10, 100, 500, 1000 and 1500 clients to illustrate the models behavior under different load conditions. The TPC-C transaction mix is that of Table 1. Figures 4 - 6 show the throughput, mean response time, mean queuing time for the TPC-C queuing network transactions. Figures 7 and 8 show the mean queue length and mean queuing time for the tables (servers) of the TPC-C queuing network model. Figure 9 shows the relationships of the mean queuing times on each table for each TPC-C transaction.

As indicated by Fig. 4-6, throughput, mean response time and queuing time for the transactions increase as the load on the model is increased. This holds for the mean queue length and mean queuing time for the tables from Fig 7 - 9. This result is typical of any database system: the transaction load affects performance.

Using these figures, we will demonstrate how a database designer can deduce performance indications from the database design.

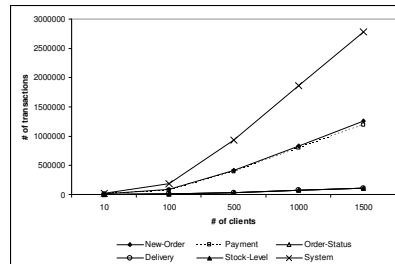


Fig. 4. Simulation results for the throughput of the TPC-C transactions.

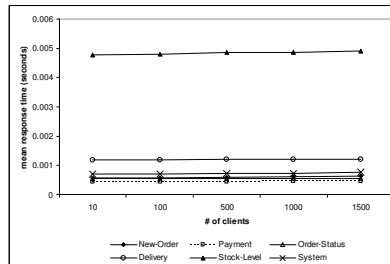


Fig. 5. Simulation results for the mean response time for the TPC-C transactions.

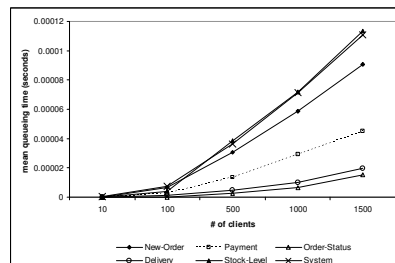


Fig. 6. Simulation results for mean queuing time of the TPC-C transactions.

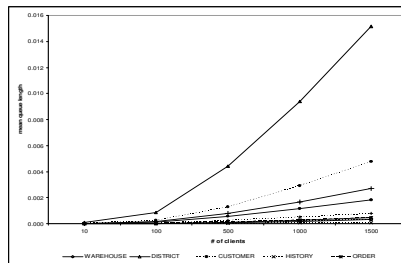
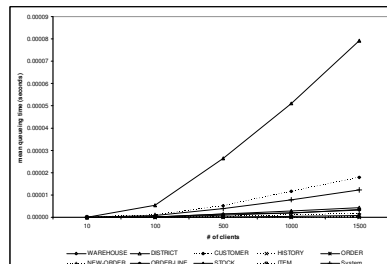


Fig. 7. Simulation results for the mean queue length for the TPC-C tables.

Fig. 8. Simulation results for the mean queuing time for the TPC-C tables.



From Fig. 5 and 6, the Stock-Level transaction has the longest mean response time and mean queuing time, longer than the overall system mean response and queuing times, even though the Stock-Level transaction has low throughput (Fig. 4), 4% of the total throughput (this is incorporated in the TPC-C benchmark specifications). This indicates that this transaction is a candidate for performance tuning, i.e. redesign.

In addition, from Fig. 7 and 8, the STOCK table has the longest mean queue length and the longest mean queuing time, longer than the overall system mean queuing time, even though the STOCK table is accessed by only two transactions (New-Order and Stock-Level), in contrast to the CUSTOMER table that is accessed by four transactions (New-Order, Payment, Order-Status, Delivery). Furthermore, from Fig. 9, the majority of the waiting time spent by the New-Order and Stock-Level transactions is queuing for the STOCK table. Given that the TPC-C benchmark specifies the New-Order transaction as the measure of system performance [18], the STOCK table is a major bottleneck for the New-Order transaction as well as the Stock-Level transaction.

The data retrieved by the New-Order and Stock-Level transactions from the STOCK table cannot be changed due to the TPC-C specifications, i.e. these transactions cannot be redesigned and hence their service demands cannot be changed. Therefore, a redesign of the STOCK table or its access methods (indexes) would benefit the response time of both the Stock-Level and New-Order transactions. This conclusion is consistent with the TPC-C implementations that have kept the STOCK table resident in the DBMS buffer [19-23], therefore eliminating I/O disk access, thus decreasing transaction response times.

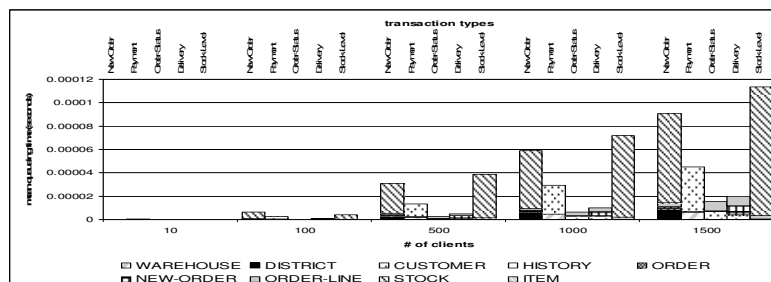


Fig. 9. Simulation results showing the relationships between mean queuing times on the tables for the TPC-C transactions for different number of clients.

4 Conclusions & Future Work

In this paper, we have proposed a novel modelling technique to evaluate the performance of database designs using queuing networks. This original approach adds the element of dynamic modelling of the database design, giving the database designer more visibility on the expected performance of the design before implementation, thereby improving database designs and preventing costly post-deployment database tuning.

We have shown that our database design performance model has the ability to evaluate expected database system performance from database designs. This has been established through the modelling of the TPC-C benchmark design and comparing the simulated results with the disclosed TPC-C benchmark implementation results. The queuing network model for the TPC-C benchmark database design was able to

capture transaction performance behavior and to pinpoint database design artifacts for performance redesign.

This work is an early step towards a performance evaluation methodology for database designs. Extensions to the current database design performance model are proposed by adding a more detailed representation of DBMS, namely a refined representation of the DBMS storage subsystem and the incorporation of temporary in-memory SQL operations in the query I/O cost model.

With the emergence of 24x7 Web-based e-service applications with back-end databases performance evaluation of database designs can no longer be ignored.

References

1. Thomasain, A.: Performance analysis of database systems. In: G. Haring, C. Lindemann, Reiser, M. (eds.) *Performance Evaluation: Origins and Directions*. LNCS, vol. 1769, pp. 305-327. Springer-Verlag, Heidelberg (2000)
2. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: a survey. *IEEE Transactions on Software Engineering*. 30, 295-310 (2004)
3. Pressman, R.S.: *Software engineering: a practitioner's approach*. McGraw-Hill Higher Education, New York; London (2005)
4. Smith, C.U., Williams, L.G.: *Performance solutions: a practical guide to creating responsive, scalable software*. Addison-Wesley, Boston, MA; London (2001)
5. Sevcik, K.C.: Data Base System Performance Prediction Using an Analytical Model. In: *Proc. VLDB'81*, pp. 182 -198. IEEE Computer Society (1981)
6. Adams, E.J.: Workload models for DBMS performance evaluation. In: *Proc. of the Thirteenth ACM Annual Conference on Computer Science*, pp. 185 -195. ACM Press (1985)
7. Casas, I.R., Sevcik, K.C.: Structure and validation of an analytic performance predictor for System 2000 databases. *Information Systems and Operational Research (INFOR)*. 27, 129-144 (1989)
8. Hyslop, W.F., Sevcik, K.C.: Performance prediction of relational database systems. In: *Proc. of the Canadian Computer Measurement Group (CMG) Conference* (1991)
9. Salza, S., Tomasso, R.: A modelling tool for the performance analysis of relational database applications. In: *Proc of 6th Int'l Conf. on Modelling Techniques and Tools for Computer Performance Evaluation* (1992)
10. Menascé, D.A., Gomaa, H.: A method for design and performance modeling of client/server systems. *IEEE Transactions on Software Engineering*. 26, 1066-1085 (2000)
11. Osman, R.I.M., Awan, I.U., Woodward, M.E.: A framework for the performance evaluation of database designs. In: *The 23rd Annual UK Performance Engineering Workshop (UKPEW 2007)* (2007)
12. Shasha, D., Bonnet, P.: *Database tuning: principles, experiments, and troubleshooting techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
13. Ramakrishnan, R., Gehrke, J.: *Database management systems*. McGraw-Hill, Boston, Mass. (2003)
14. Kifer, M., Bernstein, A.J., Lewis, P.M.: *Database systems: an application-oriented approach*. Pearson/Addison Wesley, Boston (2005)
15. Zilio, D.C., Rao, J., Lightstone, S., Lohman, G.M., Storm, A., Garcia-Arellano, C., Fadden, S.: DB2 Design Advisor: Integrated Automatic Physical Database Design. In: *Proc. VLDB'04*, pp. 1087-1097. (2004)
16. Elmasri, R., Navathe, S.B.: *Fundamentals of database systems*. Addison-Wesley (2007)
17. Hsu, W.W., Smith, A.J., Young, H.C.: I/O reference behavior of production database workloads and the TPC benchmarks-an analysis at the logical level. *ACM Trans. Database Syst*. 26, 96-143 (2001)
18. Transaction Processing Performance Council: TPC benchmark C: standard specification, revision 5.8.0. (2006), http://www.tpc.org/tpcc/spec/tpcc_current.pdf

19. Hewlett-Packard: TPC benchmark C: full disclosure report for HP ProLiant ML350 G5 using Oracle Database 10g Standard Edition one and Oracle Enterprise Linux 4, second edition. (2007), http://www.tpc.org/results/FDR/TPCC/HP_ML350G5_080831_FDR.pdf
20. Hewlett-Packard: TPC benchmark: full disclosure report for HP ProLiant ML350 G5 using Oracle Database 11g Standard Edition One and Windows 2003 SP1 R2. (2007), http://www.tpc.org/results/FDR/TPCC/HP%20ML350G5_Oracle_070912_FDR.pdf
21. IBM: TPC benchmark: full disclosure report for IBM System p 570 Model 9117-MMA using AIX 5L version 5.3 and Oracle Database 10g Enterprise Edition, first edition. (2007), http://www.tpc.org/results/FDR/TPCC/IBM_570_4_20070806_fdr.pdf
22. IBM: TPC benchmark: full disclosure report for IBM System p5 570 using Oracle Database 10g Rel 2 Enterprise Edition and Red Hat Enterprise Linux Advanced Platform 5 for POWER, first edition. (2007), http://www.tpc.org/results/FDR/TPCC/IBM_P570_Linux_Oracle_071005_FDR.pdf
23. Bull: TPC benchmark: full disclosure report for Bull Escala PL1660R using AIX 5L version 5.3 and Oracle Database 10g Enterprise Edition, first edition. (2007), <http://www.tpc.org/results/FDR/TPCC/Bull-Escala-PL1660oracle-FDR.pdf>
24. Potier, D.: New users' introduction to QNAP2. INRIA (1984)
25. Feuerstein, S.: Oracle PL/SQL Programming Guide to Oracle8i Features. O'Reilly & Associates (1999)
26. Feuerstein, S.: Oracle 10g Adds More to FORALL. Oracle Magazine (2004)
27. Schrag, R.: How Bulk Binds in PL/SQL Boost Performance. (2005), http://www.dbspecialists.com/presentations/bulk_binds.html

Appendix: Modelling Assumptions

1. SQL Cost Estimation Assumptions
 - a. SQL statement DB pages cost was based on the cost model described in [13].
 - b. Temporary tables are completely held in main memory; therefore any manipulation of intermediate results incurs no costs.
 - c. It is assumed that all data pages are flushed from memory after a SQL statement completes its operations on the data; no caching is involved.
 - d. Relational algebra JOIN operations are modelled on the queuing network as sequential access to the tables in the order that they are accessed by the query optimizer (this information is available on the query tree).
2. TPC-C Benchmark Assumptions
 - a. No transaction rolls back;
 - b. The Payment and Order-Status transactions are invoked using the customer's last name only;
 - c. The effect of the growth of the tables due to the execution of the New-Order transaction is not taken into account when calculating service demands for transactions.
 - d. The average value for all parameters is used.
 - e. The average number of customers with the same last name and the average number of repeated items in 20 orders for the Stock-Level transaction was calculated through a simulation of the nonuniform random functions stated in the benchmark with parameter C=1.
 - f. Time to return data to the client during the execution of the transaction is not considered.
3. Miscellaneous Assumptions
 - a. Database block size=2048 bytes and DB pages are fully loaded.
 - b. Disk I/O access time is 0.00002 seconds per DB block/page.
 - c. Locking and locking delay is not considered.

Construction of novel continuous time Markovian queues for exact solution

David J. Thornley*

Abstract

Modern, complex computation and network traffic patterns demand sophisticated modeling representations which capture detail of both timing and magnitude of processing and communication activity. Batched queuing models aim to achieve this by including stochastic unit and non-unit transitions between queue lengths at instants generated by Markov modulated Poisson processes. It has previously been shown that an MMPP may approximate a general interval distribution to arbitrary precision. We now describe how a general batch size distribution may be approximated using superposition of compound Poisson processes with real and complex distribution parameters, these latter taking the form of imaginary distributions, which sum to give real, positive distributions. We explore the construction of probability flux patterns in the continuous time Markov chain which can be used to approximate general batch transaction models. This may be preferable to the heuristic solution spectrum truncation common when analysing arbitrary batch transaction size distribution with a more traditional matrix analytic approach.

1 Introduction

The continuous time, discrete state Markov modulated queue is one of the most thoroughly research performability modeling entities. Time intervals in real systems can be matched by using phase-type distributions, or more general Markov arrival processes, each of which synchronizes a transition in the queue with absorption in a transient Markov chain. Such distributions can approximate any distribution given sufficient states in the chain. The the gap between provision of this capacity and implementation in a practical system lies in the establishment of an effective and stable matching approach, many of which have begun to emerge recently.

Éltető, Rácz and Telek have described the use of minimal coefficient of variation matrix exponentials in the approximation of deterministic time intervals [8]. While the particular approach to generation of these continuous distribution approximations has not yet been proven, a small class has been successfully formalized more recently [11], and the value of such results is beyond question. They describe modal and oscillatory terms which arise from the eigensystem of the matrix, and these can be used to build interesting distributions if a suitable matrix can be formulated with the required eigenvalues.

*Department of Computing, Imperial College London, 180 Queen's Gate, South Kensington, London SW7 2RH, djt@doc.ic.ac.uk

In addition to a requirement for control over time interval distributions, it is also desirable to match real distributions of transaction sizes, commonly called batches. We may require large, deterministic batch transaction sizes in a queue, while retaining efficient solubility. We have previously formulated a means by which queues with geometrically distributed batch sizes may be solved, and this is achieved in a directly soluble discrete state, continuous time Markov modulated queue which already involves mathematical entities closely related to those discussed by Éltető *et al.* The present work relates to the formulation of queues based on geometrically distributed batch transaction sizes. These are discrete state, continuous time Markov modulated queues in which the probability of transitioning between two queue levels is a sum of terms geometric in the jump size. We will refer to this simply as a geometrically batched queue. The novel formulation in batch size distribution provides the means by which arbitrary distributions may be constructed. It is well known that any waveform may be approximated by a superposition of a number of sinusoids (consider the work of Fourier, and the discrete cosine transform in signal analysis work). Since we provide oscillatory terms, this introduces a similar degree of flexibility in transaction size distribution to that introduced in time interval by Markov modulation. There are also opportunities for coarse approximation using simpler modal terms.

In this paper we describe the formulation and preparation for solution of a class of queues with batched arrival, departure and removal processes defined on bands of queue lengths, with parameters chosen to reproduce non-monotonically distributed batch sizes. We add to the state of the art by introducing batched processes to banded queues, and by identifying and quantifying the opportunity to produce non-monotonic batch size distributions.

2 Background

Work on the construction of geometrically batched queues for direct solution (in contrast to a maximum entropy approach introduced by Kouvatsos [13]) took off with an ATM model by Harrison and Chakka [15], with exact solutions using spectral expansion as espoused by Mitrani and Chakka [12]. This was later developed to incorporate a negative customer arrival process [16]. This important step introduced the concept of superposition of departure processes in the processing completion and customer removal due to negative customers. Chakka and Harrison then provided proofs of the correctness of localized equations for a canonical geometrically batched queue [17], and with positive and negative customer arrivals [4]. In the same year, Harrison's MEGAN research project to investigate the use of geometrically batched queues in networks began. Harrison's concept of a geometrically batched node with superposed arrival streams and link traffic approximated as a compatible process led to a successful implementation of a processor-farming network model with multiple queues and feedback [10]. As part of this process, the formulation approach for the queue itself underwent a radical change, resulting in an automated process which will formulate finite balance equations for a geometrically batched queue with an arbitrarily complex description [20].

Concurrently with our dissemination of this automated approach, Chakka *et al* provided important example applications, with the addition of a more com-

plex processing description, using the original approach of hand-crafting the equations [2, 3, 6]. We successfully implemented an important enhancement to the queue formulation – proposed by Chakka – in which queue length dependency is approximated by defining processes in bands of queue length, with a clear demonstration of the improvement of accuracy by comparison with simulation [18] using geometrically batched queues in a network with tight feedback to emphasize the value of the banding. Do, Chakka *et al* have used an *unbatched* banded queue to model a web service [5]. We believe this to result from the prohibitively high complexity of their preferred approach to balance equation localization. We present here an imperative approach which represents an additional simplification of the recursive approach we first presented in [20].

We, in common with Chakka and co-workers, have consistently used spectral expansion as originally advocated by Mitrani and Chakka [12] due to its explicit presentation of the independent components of the behaviour of the solution. The controversy surrounding this decision - which may have contributed to the delay in the material from [2] reaching a journal [1] in an updated form - has been resolved [19]. The choice of solution mechanism is now open, and depends more on the precise form of the queue, and the context within which it is to be used. Chakka [1] explains the spectral expansion method clearly, so we refer to this with some addition of detail relating to the solution of banded queues.

3 Probability flux geometrically distributed over jump size

The present development of the automated approach enables us to view the geometric terms in a more abstract manner. In [1], for example, the queue formulation is prepared in terms of superposed streams of customer transactions. It is also reiterated that this superposition can act to approximate hyperexponential distributions. However, when we view the batch transaction process, we note that the single requirement for validity is that the total probability of transition is constant. This frees the use of the geometric terms to be a basis set for constructing any distribution we desire, so long as the result is always positive. This is exactly the same premise as used by T. Éltető *et al* when describing freely defined time delays constructed from the eigenvalues of the matrix exponential. In that case, the problem of matching a real process is complicated by the freedom of expression in the matrix exponential. In our case, we may directly state the components. These may be complex conjugate pairs to evoke a real oscillatory term, or negative to contribute to a modal distribution.

One advantage of this approach is that we now have a method for constructing a class of queues for which there is a wide range of results relating to measures of sojourn time and reward, which can be extended to encompass the new formulation. Another is that this work complements recent forays into alternative representations for queues, including the Fokker Planck equations [7], by bringing us closer to matching arbitrary network node work patterns - in terms of both time delay and task magnitude. An important consequence of the simplified, mechanical process we describe for providing finite Chapman Kolmogorov probability flux balance equations is the ability to rapidly prototype novel queues which can be solved for equilibrium state occupation probabilities,

and hence analysed for further measures, and incorporated into networks.

We have recently analysed the character of the eigenspectrum of the equilibrium solutions for Markovian queues [19], revealing that the spectrum size can be controlled if a certain constraint is placed on the formulation of the queue. If the transition processes between queue lengths for any given parameter selection phase comprise a sum of geometric series in the jump size, then the spectrum size is finite, and can be calculated based on the queue description before solution is attempted. We do not claim that this is the only satisfactory constraint, but it has proven sufficient. This constraint is satisfied by the class of geometrically batched Markov modulated Poisson process queues.

4 Queue balance equation formulation

The total steady state vector of a queue, dotted with the j^{th} column of the instantaneous generator matrix of the queue is commonly shown in the following form:

$$\sum_{i=1}^f \mathbf{v}_{j-1} p F_i + \mathbf{v}_j p(L) + \sum_{i=1}^b \mathbf{v}_{j+1} p B_i, \text{ where } L = Q - D \left(\sum_{i=1}^f F_i - \sum_{i=1}^b B_i \right)$$

Where \mathbf{v}_j is the vector of state occupation probabilities at queue length j for each modulation state. When this term is set equal to a vector of zeros of the appropriate length, it provides a Chapman Kolmogorov balance equation for the queue's steady state. Throughout this paper, we omit showing the equation of such probability flux terms to a zero vector.

Matrices F_i hold the probability flux rates for a jump size of i due to arrivals. Similarly, B_i give processing completions or removals (for example, due to negative customers). These matrices are not always diagonal, for example when phase type or MAP processes are used. To introduce our particular formulation of such a queue, which is less general because of the constraint on the form of the matrices F and B , but guaranteed soluble with a finite eigenspectrum of calculable size. In [19] we prove that the eigenspectrum size can be a barrier to solubility. We consider a general formulation of the balance equations within the class of geometrically batched queues.

Let us consider three main forms of geometric component for constructing batch transaction size distributions, in common with much previous work on the geometric queue. We reproduce the expression from [20] which defines the Chapman Kolmogorov probability flux term for any queue length in an MM CPP_k/GE_k/kc/L G_k queue (*i.e.* k processors of same rate):

$$\begin{aligned} p_j = & \sum_{i=0}^{j-1} \mathbf{v}_i \left[\sum_{k=1}^{n^{arr}} \Lambda_k (1 - \Theta_k)^{f_{j < L}} \Theta_k^{j-i-1} \right] \\ & + \mathbf{v}_j \left[Q - \sum_{k=1}^{n^{arr}} \Lambda_k f_{j < L} \right. \\ & \quad \left. - \sum_{k=1}^{n^{kill}} K f_{((j > \kappa^b) \vee h^p)} \beta_j - \sum_{k=1}^{n^{serv}} C_{k,j} \right] \\ & + \sum_{i=j+1}^L \mathbf{v}_i \left[\sum_{k=1}^{n^{kill}} K_k (1 - R_k)^{f_{j > \kappa^b}} R_k^{i-j-1} f_{\substack{(j \geq \kappa^b) \\ \vee (h^p \wedge i=j+1)}} \beta_i \right. \\ & \quad \left. + \sum_{k=1}^{n^{serv}} C_{k,i} (1 - \Phi_k)^{f_{j > c-1}} \Phi_k^{i-j-1} f_{\substack{(i=j+1) \\ \vee (j \geq c-1)}} \right] \end{aligned}$$

This describes the probability flux at any queue length. We now describe the formulation of this expression, progressively introducing the terms. We see that the queue can accommodate geometrically batched occurrences of both arrivals and processing completions. Formally, we have transitions within the queue whose vertical component (increasing or decreasing queue length) has size s with probability $(1 - \theta)\theta^{s-1}$, and the horizontal component is zero. As with the rates Λ and M , we write the batch parameters in matrix form, where Θ is the diagonal matrix of geometric arrival batch size parameters (θ) and Φ similarly for service batch size parameters (ϕ) .

Each processor's Poisson completion intervals can be given by a matrix M , with batching described by the matrix Φ . To incorporate multiple homogeneous processors, the processing rate matrix M is replaced by $C_j = \min(j, c)M$ at queue length j , with the batch matrix Φ remaining unchanged. If all processors are busy, then service batches can clear jobs down to level $c - 1$ inclusive; *i.e.* there is *unbounded* batch flow to levels c and above, and *bounded* (truncated) batch flow to $c - 1$. If any processors are idle, then the processing batch size is exactly 1, as there are no jobs in the waiting room, and a processor can only clear its own job in service.

We use a switching term $f(P)$ which is equal to 1 if predicate P is true, and zero otherwise. The switch $f_{j>c-1}$ bounds downward flow at level $c - 1$. The term $f_{(i=j+1) \vee (j \geq c-1)}$ selects valid flows, which are batches from anywhere in the waiting room to just below c , or single jobs from a single processor.

We treat breakdowns and repairs (in the sense of Mitrani and Chakka [14]) by allowing the number of processors to vary from 0 to c across the modulation phases. Thus the number of phases associated with the processing description is $N = c + 1$. This is introduced into the left-hand side expression by replacing references to c with the vector (c_1, \dots, c_N) of the numbers of operational processors in each modulation state.

The switching term $F_{P(m)}$ is a diagonal matrix of values whose i^{th} diagonal element is $f_{P(i)}$. We define the result of raising a square matrix A to the power B with the same dimensions to be a similar matrix of elements $a_{i,j}^{b_{i,j}}$. (In fact all matrices operated upon here are diagonal.) Also, the m^{th} element of C_j is now $\min(j, c_m)\mu$, where $c_m = (m - 1)$. We combine breakdowns and repairs with modulated arrivals by the standard technique of taking the Kronecker product of the independent modulation matrices.

Rate matrices with off diagonal elements describe simultaneous queue length changes with queue modulation phase transitions. These are used to create phase-type and MAP processes.

Negative customers create additional probability flux downward in the lattice, corresponding to queue length decreases due to customer loss. This is used to model network phenomena such as losses and to approximate load balancing. As well as specifying the Poisson rate and batch size parameters in matrices K and R respectively, a killing mode has to be chosen. We consider three modes: t^v or "tail vulnerable" removes a job from the tail of the queue even if it is in service, t^s or "tail safe" removes a job from the tail of the queue but not when in service, and h^p or "head per" which removes a customer from the head of the queue (in service) at an independent but equal rate per processor, leading to a lower loss rate when some processors are inactive.

Killing mode t^v is the simplest, as it can kill any job in or out of service,

and the batches are bounded only at level zero. Mode t^s is bounded at level c , as it cannot kill any job in service. Mode h^p causes flux identical to processing completions.

Queue length κ^b is the lowest level reachable by killing, *i.e.* $\kappa_m^b = c_m f_{t^s} + (c_m - 1)f_{h^p}$, at which batch killing is truncated. We define the m^{th} diagonal element of the killing factor matrix at level (queue length) j , $\beta_j = \min(j, c_m) / \max_m(c_m)$ for h^p killing and $c_m / \max_m(c_m)$ otherwise.

To add multiple processes, we augment the arrival term to reflect a sum of streams, for example n^{arr} is the number of arrival streams. This introduction of sums of streams is also performed for processing completions and negative customers to create a more general balance equation.

Queue length dependency can be approximated by defining the component processes according to bands of queue length. Every term in the flux expression can be indexed according to its band to select the appropriate entries. Note that this includes Q the modulator. Each such band exhibits its own repeating region with a distinct eigensystem. The simplest way to add this to the expression for the balance equation is to modify the term indices to include the queue length from which their probability flux is sourced. In the following analysis, we also allow for the presence of finite jumps in the system.

5 Localization

Geometrically distributed customer batch sizes are unconstrained in size, so the matrix recurrence relation in the balance equations for a queue using them can be of infinite order. To enable the solution of such problems using spectral expansion or matrix geometric methods, we introduce a novel variant of an algorithm introduced in [20] that automatically transforms the problem into an equivalent whose repeating region is described by finite probability flux balance equations. We refer to these transformed equations as *localized equivalent balance equations*.

Localization of these equations is achieved using a similar trick to that employed in finding the value of an infinite geometric series, namely taking the difference between the series itself and a copy scaled by the geometric factor. Terms differing by the appropriate factor are found in the coefficients of probability flux terms for neighbouring queue lengths. The range of resulting equations arises from the pollution of the otherwise geometric series – which are eliminated – by any constant terms around the focus of the balance equation (such as the modulator’s instantaneous generator equation), and the innermost terms of the series persisting from previous eliminations.

When we eliminate the terms arising from other bands, the geometric series in the coefficients of the vector SOPs begin at a given queue length, with the result that there are no remnants. Thus, it is not necessary to write down terms from a neighbouring band, except in balance equations which will span a

boundary between bands when localized.

$$\begin{aligned}
p_{j-1} &= \dots \mathbf{v}_{j-3}[F_2 + \Lambda(I - \Theta)\Theta] \\
&\quad + \mathbf{v}_{j-2}[F_1 + \Lambda(I - \Theta)] \\
&\quad + \mathbf{v}_{j-1}[L] \\
&\quad + \mathbf{v}_j[B_1] \\
&\quad + \mathbf{v}_{j+1}[B_2] \dots \\
p_j &= \dots \mathbf{v}_{j-3}[F_3 + \Lambda(I - \Theta)\Theta^2] \\
&\quad + \mathbf{v}_{j-2}[F_2 + \Lambda(I - \Theta)\Theta] \\
&\quad + \mathbf{v}_{j-1}[F_1 + \Lambda(I - \Theta)] \\
&\quad + \mathbf{v}_j[L] \\
&\quad + \mathbf{v}_{j+1}[B_1] \dots \\
p_{j-1}\Theta - p_j &= \dots \mathbf{v}_{j-3}[F_2\Theta - F_3] \\
&\quad + \mathbf{v}_{j-2}[F_1\Theta - F_2] \\
&\quad + \mathbf{v}_{j-1}[L\Theta - F_1 - \Lambda(I - \Theta)] \\
&\quad + \mathbf{v}_j[B_1\Theta - L] \\
&\quad + \mathbf{v}_{j+1}[B_2\Theta - B_1]
\end{aligned}$$

Note that the geometric terms have disappeared, leaving only the constant terms unrelated to the geometric series of Θ scaled and mixed. We now show the removal of processing completion batched from a separate band at higher queue lengths. We simplify the presentation by using the case where we have only unit constant transitions – or one batched process already localized in either direction – other than the batches from the band. These are un-indexed for clarity.

$$\begin{aligned}
p_j &= \mathbf{v}_{j-1}[F] + \mathbf{v}_j[L] + \mathbf{v}_{j+1}[B] \\
&\quad \dots + \mathbf{v}_{j+k}[M(I - \Phi)\Phi^{k-1}] \\
&\quad + \mathbf{v}_{j+k+1}[M(I - \Phi)\Phi^k] \dots \\
p_{j+1} &= \mathbf{v}_j[F] + \mathbf{v}_{j+1}[L] + \mathbf{v}_{j+2}[B] \\
&\quad \dots + \mathbf{v}_{j+k}[M(I - \Phi)\Phi^{k-2}] \\
&\quad + \mathbf{v}_{j+k+1}[M(I - \Phi)\Phi^{k-1}] \dots \\
p_j - p(j_1)\Phi &= \mathbf{v}_{j-1}[F] + \mathbf{v}_j[L - F\Phi] + \mathbf{v}_{j+1}[B - L\Phi] \\
&\quad + \mathbf{v}_{j+2}[-B\Phi] \text{ with no peripheral terms}
\end{aligned}$$

This transformation is simple matrix arithmetic. Each such transformation employing a pair of flux descriptors removes a single geometric series. Therefore to remove a number of geometric series, we perform this elimination a number of times. Since each flux descriptor contributing to an elimination may itself be the result of one or more eliminations, this process is recursive. This is clearly seen in [20], in which we provide a recursive operator which interrogates the algebraic flux descriptor to generate elimination factors. We have since realised that the problem is rather simpler if we construct the algebraic expressions ourselves, since we know the Θ , Φ and other such geometric terms *a priori*. In essence, we use the form $p_{j-1}\Theta - p_j$ to eliminate upward flux from a flux expression

at queue length j , and the form $p_j - p_{j+1}\Phi$ to eliminate downward flux terms. We only differentiate between the geometric terms Φ as a visual reminder of the direction of the flux involved. Recall that these flux expressions are to be equated to zero in the solution to the queue, so these transformations are full rank as described because it is a simple column operation (recalling that we are working with left vectors).

Aside: it is interesting to note that the “rate” terms from exogenous band geometric terms do not appear in the localized expressions. However, the rate terms from all the equations are linked by their appearance in the use of explicit balance equations at boundaries between bands.

5.1 Parameters of the localization

In this glossary of definitions, an upward process increases queue length, and a downward process decreases it. For all diagonal matrix terms, appending an additional subscript creates a reference to the corresponding diagonal element.

r is the number of bands in the queue *above* the processor filling region, which we refer to as the queue *resolution*. Bands are thus indexed on b , $0 \leq b \leq r$.

k_b is the queue length at the top of band b . Bands are numbered from zero. The zeroth band is by our convention the processor filling region.

d_b is the number of infinite geometric series associated with downward processes defined by band b .

u_b is the number of infinite geometric series associated with upward processes defined by band b .

$\Phi_p^{(b)}$, $1 \leq p \leq d_b$ are the geometric term matrices for downward processes defined by band b . These may or may not have been packed as described later.

$\Theta_p^{(b)}$, $1 \leq p \leq u_b$ are the geometric term matrices for upward processes defined by band b . These may or may not have been packed as described later.

$u_b^{(l)}$ is the maximum range of finite local upward transitions defined in band b . It is essential to remember that these will cross band boundaries.

$d_b^{(l)}$ is the maximum range of finite local downward transitions defined in band b . Again, these will also cross band boundaries.

$u_b^{(*)}$ is the total number of geometric series defined in independent matrices (which may or may not have been packed as defined below) in band b and from all lower bands which land in band b . This excludes finite local transitions.

$d_b^{(*)}$ is the total number of geometric series defined in independent matrices (which may or may not have been packed as defined below) in band b and from all higher bands which land in band b . This excludes finite local transitions.

These terms provide the necessary measurement to correctly select the number of balance equations required in the elimination process, the number of coefficients to enumerate and process, and hence to predict the range of queue lengths involved in the resulting localized balance equation, and the number of associated eigenvalues in the solution.

5.2 Preparing for localization

We outline the procedure for mechanically generating a balance equation for queue length j . Begin by populating an array A with the matrix coefficients of \mathbf{v}_{j-x} through \mathbf{v}_{j+y} which appear in the unlocalized probability flux vector

expressions for queue lengths $j - u$ through $j + d$. $x + 1 + y$ is therefore the range of queue lengths covered by the transformed matrix balance equation due to locally generated and exogenously generated infinite geometric series, and local finite jumps. From the terms detailed above, $x = u_b^* - f(j = k_{b-1} + 1)u_b$, $y = d_b^* - f(j = k_b)d_b$ and $u = x + u^l$, $d = y + d^l$, where $f(P)$ is a switching term of the same form used in the definition of the balance equations, equal to one if P is true, and zero otherwise.

We suggest that the columns pertain to a given balance equation so that they resemble the example eliminations we provide above. This means that the array resembles a rectangular excerpt from the instantaneous generator matrix for a whole queue commonly depicted in works relating to the matrix analytic approach. So, $A_{i,j}$ is the coefficient of \mathbf{v}_i in the raw balance equation for queue length j .

This procedure does not explicitly acknowledge the presence of processes defined on bands of queue length. These are encoded entirely in the balance equation expression - for example that given earlier for p_j - and the calculation of a number of terms, including the position and extent of the elimination window (the array we populate with coefficients), and the provision of an array of diagonal matrices signifying the geometric terms.

5.3 Compaction of problem parameters

This elimination process allows for a system in which all the diagonal elements of the geometric terms are non-zero. Any zero values reduce the span of the equations in the corresponding modulation state, and reduce the number of eigenvalues in the solution. If any pair of values in a given modulation state in separate geometric terms in a given direction (increasing or decreasing queue length) is identical, this also reduces the span of the equation and hence number of associated eigenvalues in the system, since the elimination of one also knocks out the other. If there is a reduction of the number of independent non-zero terms in all modulations states, then the localized balance equation will involve fewer queue lengths over all. To simplify the analysis of the system in terms of number of eigenvalues and the range of the balance equations, we compact the representation of the balance equation.

Create fresh copies of all the batch distribution matrices in groups by band of queue lengths. We are going to separate the definition of the general balance equation from the description of the localization. We use separate copies of the batch distribution terms. This is of course potentially spacially inefficient, but it makes the process simpler in implementation, and simpler to understand.

If the queue we wish to formulate uses a number of modulated processes selected by an environmental modulator (for example BMAP arrivals and PH type processing), the rate matrices for each will have a number of zero rows. The associated geometric batch distribution parameter matrices therefore also have these zero rows.

The copies of the matrices to be used in the elimination process will either be numerical or symbolic. If they are symbolic, then any elements known to be non-zero should be references to the corresponding elements in the original matrix.

For each band, therefore, we have a list of matrices describing the geometric terms generated by processes in that band. To enable convenient prediction

of the number of eigenvalues in the system, we need to know what the per-modulation-state range of the localized balance equations will be. To find this, we *compact* the set of distribution matrices.

The simplest view of this is, for each flux direction separately, to write the diagonals of each matrix into successive columns of a array. We then shift each element of that array to the leftmost column possible. This may leave some all-zero columns. Each number of non-zero elements in row m of the array produced for upward transitions gives the value of $u_{b,m}$. We count the corresponding elements in the array for downward processes to give $d_{b,m}$.

5.4 Localization procedure

The simplest implementation results from removing all the series in one direction, then removing those from the other. Here we remove all backward geometric processes, followed by the forward. In C-like pseudocode:

```
// number of columns in the matrix is N
N = num_up_terms + 1 + num_down_terms;

// P and T are the lists of
// down and up geometric terms

// remove downward processes
for(t=1; t<num_down_terms; t++ )           // loop t geometric terms
    for(j=t; j<N; j++ )                     // loop j columns
        for(i=0; i<num_coefficients; i++)    // loop i coefficients
            set( A[i,j], sub( A[i,j], prod( A[i,j-1],P[t])));
            // this is a matrix operation

// remove upward processes
for(t=1; t<num_up_terms; t++ )             // loop t geometric terms
    for(j=N-t-1; j>=num_down_terms; j-- ) // loop j BACK columns
        for(i=0; i<num_coefficients; i++)  // loop i terms
            set( A[i,j], sub( A[i,j], prod( A[i,j+1],T[t])));
            // this is a matrix operation
```

Recall that each element of A, P and T is an array, so matrix assign (set), subtraction (sub) and multiplication (mult) functions are required. At the end of this procedure, the d_{th} column (indexed from 0) contains holds the matrix coefficients of the localized balance equation.

6 A novel interpretation of batch distribution parameters

Fackerell's thesis [9] gives a clear introduction to the use of matrix exponential distributions. Éltető, Rácz and Telek describe an important opportunity to generate feasible, meaningful distributions by unconstrained optimization through identification of a locus of minimal coefficient of variance distributions [8]. These

arise from examining the eigenvalues of the system, which can result in exponential, modal and oscillatory terms. The essential constraint is that all probabilities must be positive. We have independently noted that it is possible to construct what could be viewed as a discrete analogue of the same component terms using the formulation of the geometrically batched queue.

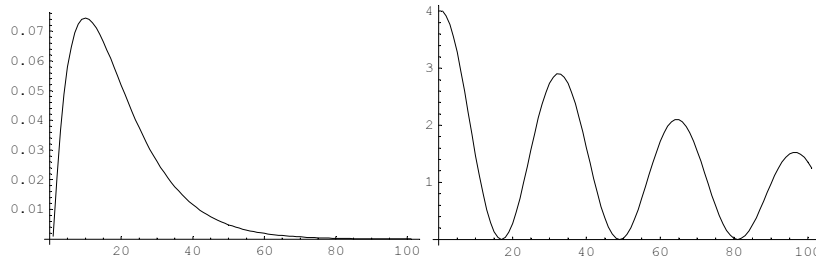


Figure 1: Left: a modal component shown as rate against jump size i for $\lambda_1 = 2$, $\theta_1 = 0.9$, $\lambda_2 = 1.999$, $\theta_2 = 0.89$. Right: a sinusoidal oscillatory component shown as rate against jump size i with $\lambda = 1$ in for the conjugate batch processes, with $\theta = 0.99e^{\pi\sqrt{-1}/16}$ in one, and θ^* in the other, and an additional real geometric term $2 * 0.99^i$ from a normal CPP process to give a positive result

A single geometric term has been demonstrated to be valuable in its own right (*e.g.* amongst others [1]). This is formed by a positive “rate” and a positive “decay”. These are commonly selected by a modulation process, so are held in matrices. Using these terms we construct the geometrically batched queues as originally conceived, with positive and negative arrivals and processing completions all geometrically batched, and Markov modulated.

We have shown how a queue with real, positive transaction size distributions expressed as a superposition of geometric terms can be solved using a novel transformation approach followed by essentially standard solution methods. We have introduced an additional aspect of novelty in the formulation of the geometric terms. Let us emphasize that the transformation process involves simple full rank manipulations which preserve the meaning of the system. This allows us to experiment freely with the geometric terms, so long as the sum is positive and real.

6.1 Modal and oscillatory components

With this freedom of expression, we can explore constructions with apparently unreasonable rates and batch size distribution values. We first introduce a *modal* term constructed of two geometric terms, one of which comprises a positive “rate” and a positive “decay”, and the other a negative “rate” of smaller absolute magnitude than the first, and a positive “decay” of smaller magnitude than the first.

$$rate_i = \lambda_1 \theta_1^i + \lambda_2 \theta_2^i, \quad \lambda_2 = -\alpha \lambda_1, \quad \alpha < 1, \quad \theta_2 = \beta \theta_1, \quad \beta < 1$$

These can be mixed and matched to form distributions with a number of modes. There is also the opportunity for an *oscillatory* component formed by a pair of

complex conjugate batch size “decays” associated with identical transaction rates, such that the complex components cancel out.

$$rate_i = \lambda((\theta)^i + (\theta^*)^i), \quad Im(\theta) \neq 0, \quad |\theta| < 1$$

These are visually comparable to the graphs in [8], with the exception that the abscissa here is discrete. The phase and frequency of the oscillatory term may be freely modified. Phase is most simply modified by adding a constant offset to the power of the complex geometric term. This preserves the necessary relationship (a constant multiplicative factor) between balance equations at different queue lengths for the elimination procedure. Note that we explicitly show the square root of minus one. The commonest name of i for this would clutter the namespace of the present work to no benefit - as soon as these oscillatory terms are recognized, we simply include a complex conjugate pair of processes, and need not consider the complex component explicitly.

These discrete geometric, modal and oscillatory components can be freely mixed, so long as the result is entirely positive. This means that there must be at least one either geometric or modal component. In initial experiments, it may be worth finding further suitable constraints in a similar sense to that espoused by Éltető *et al* for the continuous case in the matrix exponential.

We are currently exploring ensuring positivity by having one of the geometric terms specifically set to ensure the distribution is strictly positive, *i.e.* it is responsive to the evolving estimate, rather than being driven as part of the search, acting as a compensating term. This leads to shocks in the error term during convergence on distributions which have more than one minimum at or close to zero, since this incurs discontinuities in the derivatives of the compensation term. Optimization of such a fit in both the discrete and continuous case is an interesting research topic.

7 Solution

The equilibrium solution for the queue is found by setting up a matrix equation which expresses all the necessary explicit balance equations and the normalization term. These are expressed in terms of the eigensystems of the bands and any explicit SOP vectors. This is most easily built progressively by looping through the processor filling region, then through band boundaries, and the full queue if it is finite. The resulting system will be non-square due to the normalization. It is squared by either removing an equation, or adding it to another.

In the transformed system, we find that the number of eigenvalues $M = n^{up} + n^{down}$, where n^{up} is the number of distinct, geometrically batched upward transitions (customer arrivals) and n^{down} similarly the number of distinct, geometrically batched downward transitions (service completions or customer removal via negative customers) summed over modulation states.

Chakka’s solution mechanism described in [1] finds the forward and reverse eigensystems for the repeating region of the queue provided. It is not necessary to use two separate eigensystems when solving these queues, since the SOP vectors at the interface between the repeating region and a boundary may be represented explicitly in the set of equations used to finally solve the queue. Chakka aims to find the eigenvalues less than one in each process. However,

identification of the largest eigenvalue is particularly efficient. It may therefore be advantageous to use his approach of separating the processes, but solving for the maximum eigenvalues. This contrasts with the matrix geometric approach, in which a matrix embodying the smallest eigenvalues is found. Focussing on the larger eigenvalues also favours using explicit SOP vectors in the solution system for queue lengths neighbouring a band boundary. For a succinct description of use of the spectral expansion approach, we recommend Mitrani and Chakka's original work [12].

A matrix equation generated directly from the explicit balance equations involving the coefficients of the eigensystems of the bands in the queue, and the normalization term is not square. The system is over specified because of the homogeneity of the balance equations demands the addition of a normalization term. This is corrected either by removing a column of the system (recalling that we are using left vectors). We have discussed at length the relationships between spectral expansion and matrix geometric methods previously [19], providing a clear description of some limitations of each, and suggesting how they may be addressed.

7.1 Explicit regions

In general, in the region $0 \leq j \leq c - 1$, there is no repetitive structure in the balance equations that can be exploited. We therefore represent the SOP vectors for these levels explicitly as \mathbf{v}_j . To constrain these SOP vectors we generate the localized balance equations using the method in section 5 at the corresponding levels $0, \dots, c - 1$ to remove peripheral geometric processing completion terms.

For each modulation state m , the Kolmogorov equations just above the processing region, $c \leq j \leq c + u_m$, and in a finite queue or one with bands, the region just below the top of the queue or particular band, $L - d_m \leq j \leq L$, give the boundary conditions linking the explicit and repeating regions. Localization of balance equations operates as before, but the derived equations include a mixture of explicit SOP vectors represented as \mathbf{v}_j (for $0 \leq j \leq c - 1$ and $j = L$ for finite queues) or using a suitable spectral expansion or matrix geometric representation for all other levels j .

The eigensystems in separate bands are distinct, and indeed the spectrum size of each band can be different. These are joined correctly by ensuring that the solution system includes explicit equations for $u_b^{(*)}$ queue lengths below an interface between bands b and $b + 1$, and $d_{b+1}^{(*)}$ above it if the eigensystems solved for include the kernels (this requires a forward and backward system in each). However, it is far simpler to use explicit SOPs either side of the boundary and use a single eigensystem in each of the bands. This requires explicit equations for an additional queue length either side of the interface.

It is clearly desirable to allow arrival rates to vary with queue length, as this occurs naturally in a closed system. It may also be desirable to model a computation system which increases processor rate (and hence power consumption) only when a finite buffer is almost full. Banding of processing rates does not create any additional fine structure in the balance equations of the main body of the queue.

The highest band of queue lengths may be finite or infinite. If it is infinite, then there is an increased likelihood that a matrix geometric/analytic represen-

tation of the SOP vectors will be effective for that band [19]. This would neatly complement the use of spectral expansion in the internal finite bands, which motivated by accuracy arguments [19], with the potential for higher efficiency. For example, in the queues used in [18], we used spectral expansion in both bands as the queues are finite. That work does not include Markov modulation - instead, it focuses on the effectiveness of banding by exploring a network in which batch sizes are very large, effected by a process completion batch size distribution parameter ϕ of 0.9.

8 Conclusion

This latest development in Markovian queue formulation provides the opportunity to unify interarrival time distributions with a novel freedom in definition of batch transaction size distributions, while maintaining exact solubility. The formulation approach we have developed enables rapid prototyping of novel, complex queues. This is applicable both to analytic work, by generating symbolic balance equations in a suitable mathematical prototyping system such as Mathematica, and to implementation in performability analysis software tools where efficient numerical evaluation is key.

The formulation process is in essence arithmetic, comprising only matrix multiplication and addition. The numerical implementation for a tool may therefore straightforwardly represent the terms being manipulated as efficient tree arithmetic structures, in which the queue's parameters may be dereferenced. Thus, any balance equations automatically generated at run-time in such a software tool may be re-used for variant system conditions.

We have provided basic examples of building blocks for batch transaction size distributions. Use of these terms will enable approximation of discrete distributions such as those found in internet traffic packet sizes, for example by taking a discrete cosine transform of the required distribution and implementing the required terms with the oscillatory components provided.

Any continuous time, discrete state Markovian queue may now be augmented by the addition of controlled batch transaction size distributions and approximated queue length dependency. The opportunity to trade off solution complexity (in terms of number of geometric terms, modulation states and bands) against goodness of fit to target process descriptions enhances the class of queueing models which can provide accurate solutions in practice.

References

- [1] R. Chakka and T. Do. The MM Sigma $k=1..K$ CPPk/GE/c/L G-queue with heterogeneous servers Steady state solution and an application to performance evaluation. *Performance Evaluation*, 64(3):191–209, March 2007.
- [2] R. Chakka, T. Do, and Z. Pandi. Generalized markovian queues and applications in performance analysis in telecommunication networks. In D.D.Kouvatsos, editor, *Proceedings of the First International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs'03)*, July 2003.
- [3] R. Chakka, T. V. Do, and Z. Pándi. Exact solution for the MM-CPPk/GE/c/L G-queue and its Application to the Performance Analysis of an Optical Packet Switching Multiplexer. In *10th International Conference on Analytical and*

- Stochastic Modelling Techniques and Applications, Nottingham, United Kingdom, June 2003.*
- [4] R. Chakka and P. G. Harrison. A Markov Modulated Multi-server Queue with Negative Customers - The MM CPP/GE/c/L G-Queue. *ACTA Informatica*, 37(11–12):881–919, August 2001.
 - [5] T. V. Do, R. Chakka, O. Gemikonakli, and D. Papp. Level Dependent Band-QBD Processes Steady State Solution and Applications Working Paper. In *Second International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks, Craiglands Hotel, Ilkley, West Yorkshire, U.K. HETNETs'04*, pages WP8/1–6, July 2004.
 - [6] T. V. Do, R. Chakka, and Z. Pndi. Novel analysis method for optical packet switching nodes. In *Proceedings of the ONDM 2003 Conference, Budapest, Hungary*, February 2003.
 - [7] D. S. Dolcya, C. C. Constantinou, and S. F. Quigleya. A Fokker-Planck equation method predicting Buffer occupancy in a single queue. *Computer Networks*, 51:2198–2216, June 2007.
 - [8] T. Éltető, S. Rácz, and M. Telek. Minimal coefficient of variation of matrix exponential distributions. In *2nd Madrid Conference on Queueing Theory*, July 2006.
 - [9] M. W. Fackrell. *Characterization of matrix-exponential distributions*. PhD thesis, Adelaide University, 2003.
 - [10] P. G. Harrison, D. Thornley, and H. Zatschler. Geometrically batched networks. In *Proceedings of the Seventeenth International Symposium On Computer and Information Sciences, University of Central Florida*, October 2002.
 - [11] A. Horváth and M. Telek. On the properties of acyclic bilateral phase type distributions. In *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
 - [12] I. Mitrani and R. Chakka. Spectral Expansion Solution for a Class of Markov Models Application and Comparison with the Matrix-Geometric Method. *Performance Evaluation*, 23(3):241–260, 1995.
 - [13] D. Kouvatso. Entropy maximisation and queueing network models. *Annals of Operations Research*, 48:63–126, 1994.
 - [14] I. Mitrani and R. Chakka. Spectral expansion solution for a class of markov models Application and comparison with the matrix-geometric method. *Performance Evaluation*, 23:241–260, 1995.
 - [15] P.G.Harrison and R.Chakka. The MMCPP/GE/c queue as a Node Model for ATM Networks. In *Proceedings 12th UK Performance Engineering Workshop, Edinburgh*. University Press, 1996.
 - [16] P.G.Harrison and R.Chakka. The Markov Modulated CPP/GE/c/L Queue with Positive and Negative Customers. In *Proc. 7th Int. Conf. on Performance Modeling and Evaluation of ATM Networks*, 1999.
 - [17] R.Chakka and P.G.Harrison. The MMCPP/GE/c queue. *Queueing Systems*, 38(3):307–326, 2001.
 - [18] D. Thornley and H. Zatschler. Analysis and enhancement of network solutions using geometrically batched traffic. In *Proceedings 19th UK Performance Engineering Workshop, Warwick*, 2003.
 - [19] D. Thornley and H. Zatschler. Exploring accuracy and correctness in solution to matrix polynomial equations in queues. In *3rd International Conference on Quantitative Evaluation of Systems (QEST 2006), University of California, Riverside, CA, USA*, September 2006.
 - [20] D. Thornley, H. Zatschler, and P. Harrison. An automated formulation of queues with multiple geometric batch processes. In D.D.Kouvatso, editor, *Proceedings of the First International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs'03)*, July 2003.

A System for Dynamic Server Allocation in Application Server Clusters

A.P. Chester J.W.J. Xue L. He S.A. Jarvis*

Abstract

Application server clusters are often used to service high-throughput web applications. In order to host more than a single application, an organization will usually procure a separate cluster for each application. Over time the utilization of the two clusters will vary, leading to variation in the response times experienced by users of the applications.

Techniques that statically assign servers to each application prevent the system from adapting to changes in the workload, and are thus susceptible to providing unacceptable levels of service. This paper investigates a system for allocating server resources to applications dynamically, thus allowing applications to automatically adapt to variable workloads. Such a scheme requires meticulous system monitoring, a method for switching application servers between *server pools* and a means of calculating when a server switch should be made (balancing switching cost against perceived benefits).

Experimentation is performed using such a switching system on a Web application testbed hosting two applications across eight application servers. The test bed is used to compare several theoretically derived switching policies. The Average Flow switching policy is shown to provide the best policy, when considering the mean response times for this application.

1 Introduction

Large e-business and e-commerce infrastructures often require multiple applications and systems. For each of these applications a separate resource must be allocated. The allocation of resources is normally conducted at the design phase of a project, through the process of capacity planning. In planning for the capacity of a system it is important to have a minimal QoS, which should represent the lowest level of acceptable service for the system. A system architecture is then developed to enable the application to support the QoS requirements.

It is possible to consider such an environment as a set of servers which is manually partitioned into clusters, with each cluster dedicated to serving requests for a specific application.

*Department of Computer Science, University of Warwick, {apc, xuewj2, liganghe, saj}@dcs.warwick.ac.uk

Internet services are subject to enormous variation in demand, which in an extreme case can lead to overloading. During overload conditions, the service's response time may grow to an unacceptable level, and exhausting the resources in this way may cause the service to behave erratically or even crash [18]. Due to the huge variation in demand, it is difficult to predict the workload level at a certain point in time. Thus, allocating a fixed number of servers is insufficient for one application when the workload level is high, whereas it is wasted resource for the remaining applications while the workload is light. Therefore, it is desirable that a hosting centre switch servers between applications to deal with workload variation over time.

Initial research in the area of dynamic server allocation has proven to be mostly theoretical, with results being provided through simulation [14]. The motivation for this work is to examine the potential for dynamic server allocation in real-world application hosting environments.

The specific contributions of this paper are:

- To report on the development of a real-world testbed for evaluating techniques for dynamically allocating servers to applications;
- To implement three server switching policies which have been theoretically derived;
- To evaluate the three implemented policies within a practical setting, and report on the pros and cons of each.

The remainder of this paper is organized as follows: Section 2 reports on related work, describing the application environments and theoretically derived switching policies. Section 3 gives an overview of the system architecture and describes the performance characteristics of an application server. Section 4 describes the process of switching servers between applications. Section 5 provides details of the experimental parameters and demonstrates the results obtained from the system. In section 6 we draw our conclusions from the results and describe the further work that we will be undertaking based on our findings.

2 Related Work

Performance optimization for single application server architectures have been extensively studied [4, 5, 7, 8, 10, 11, 13, 17, 18]. [4, 8, 13] focus on request scheduling strategies for performance optimization. In [11], the authors use priority queues to offer differentiated services to different classes of request to optimize company revenue. They assign different priorities to different requests based on their importance. [10] studies the methods for maximising profits of the best-effort requests and the QoS-demanding requests in a web farm, however, they assume static workload arrival rate in the paper. Work in [7, 17] use provisioning techniques to achieve Service Level Agreements (SLA). This research uses analytical models to analyse system capacity and allocate resources in response to workload changes to obtain guaranteed performance. Other work in [5, 18] use admission control schemes to deal with overloading and achieve acceptable performance metrics. [5] uses session-based admission control to avoid loss of long sessions in web applications and guarantee QoS of all requests, independent of a session length. [18] presents a set of techniques for

managing overloading in complex, dynamic Internet services and is evaluated using a complex web-based email service. The work in this paper focus on the scenario where multiple applications are running simultaneously in an Internet hosting centre.

Recent work [9, 12] also studies performance optimization for multiple applications in Internet service hosting centres, where servers are partitioned into several logical pools and each logical pool serves a specific application. They address the server switching issue by allowing servers to be switched between pools dynamically. [12, 14] consider different holding costs for different classes of requests, and try to minimise the total cost by solving a dynamic programming equation. The authors in [9] define a revenue function and use M/M/m queues to derive performance metrics in both pools and try to maximise the total revenue.

The work in this paper is different from [9, 12, 14] in the following respects: an actual test-bed is used in our evaluations, and thus (i) the application is not synthetic, (ii) the supporting infrastructure demonstrates the subtleties of a real-world platform, and (iii) the switching policies are implemented, feed actual system parameters, and evaluated.

3 System Overview

In this paper we consider an environment consisting of multiple applications which is deployed across a set of servers. Each of the applications considered has an identical system architecture. Modern Web application infrastructures are based around clustered, multi-tiered architectures. Figure 1 shows multiple hosted Web applications based upon the “best possible” architecture as described in [16].

The first tier in the architecture is the *presentation tier*. This comprises the client-facing web servers that are used to host static content and route requests to an available application server. The *application tier* is comprised of a static allocation of application servers which process dynamic requests from the clients, using the data persistence tier as appropriate. The *data persistence tier* is normally comprised of a Relational DataBase Management System (RDBMS) or a legacy system which is used for the purpose of permanent data storage.

In the case of a single application it is common for the presentation tier to schedule tasks across a dedicated cluster of application server machines. Strategies for request scheduling in both commercial and open-source products are generally variations on the Weighted Round Robin (WRR) strategy. The WRR approach allows for different proportions of requests to be dispatched to different application servers and, in so doing, allows some degree of support for heterogeneous server environments by allocating a higher proportion of the workload to application servers with more capacity.

Applications that require a state to be maintained throughout a user session present a significant problem for WRR strategies, as multiple requests may not be redirected to the same server. To this end several strategies have been developed to handle this scenario. *Session affinity* ensures that subsequent requests are all processed by the same application server, thus ensuring that state is maintained throughout a user session. Drawbacks to this approach are discussed in [8] and include severe load imbalances across the application cluster

due to the unknown duration of a request at the time of dispatching it to the application server, and a lack of session failover due to the single application server providing a single point of failure to the session. It is also possible for the client to store the state of the session, resubmitting it with each request. Using this approach any available application server is able to process the user's request. Similarly the data persistence tier may be used to store session data which also enables all application servers to service all requests, however this comes at the expense of increased database server/cluster utilization. These approaches are evaluated in [3]. In this paper user session data is stored on the application server that processes the initial request. Further requests are then forwarded to the same server for processing.

The multiple application environment we consider is captured by figure 1. The diagram represents the architecture for n separate applications. The main difference from the single application architecture is the conceptual view of the *set* of application servers. In our multiple application environment any of the servers available may be allocated to any of the applications either statically or dynamically. In this paper we are concerned with the allocation of servers at the application tier. Each application requires a dedicated presentation and data persistence tier.

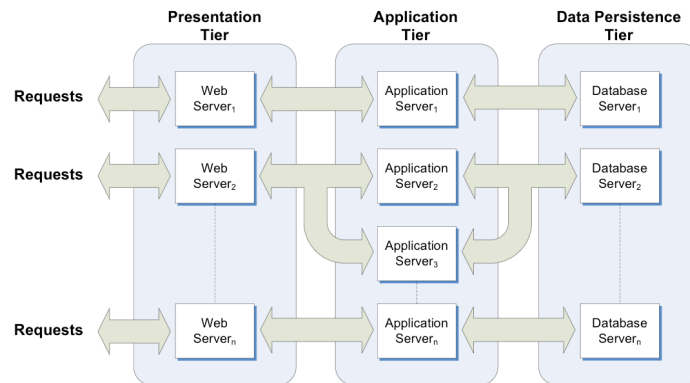


Figure 1: Multiple application architecture.

3.1 Server Performance

In [6] it is demonstrated that the throughput of an application server is linked to the number of concurrent users. While a system is under a light load with few concurrent users, the throughput of the server can increase in a near linear fashion as there is little contention for resources. As the number of concurrent users increases, the contention for system resources increases, which in turn causes the rise in throughput to decrease. The point at which the addition of further clients does not result in an increase in throughput is the saturation point, T_{max} .

From this it would follow that for a cluster of n application servers, the

maximum theoretical throughput of the cluster would be ΣT_{max} for a heterogeneous cluster. This may be simplified to nT_{max} for a cluster of homogenous servers. These theoretical throughputs are rarely achieved in practice due to the additional overheads of scheduling and redirecting requests across the cluster.

4 Server Switching

If we consider that each application hosted across the set of servers provides a service to the business (depending on the SLAs), some of the hosted applications are more important than others in terms of revenue contribution to the service provider.

Most Internet applications are subject to enormous variations in workload demand. During a special event, the visits to some on-line news applications will increase dramatically, the ratio of peak load over light load can therefore be considerable. System overloading can cause exceptionally long response times for requests or even errors, caused by the timing out of client requests and connections dropped by the overloaded application. At the same time, the throughput of the system would decrease significantly [6].

Therefore, it is desirable to switch servers from a lightly loaded application to a higher loaded application in response to workload change. In such cases, it is important to balance the benefits of moving a server to an application against the negative effects on the reduced pool and the switching cost.

The mechanism for switching servers, and the costs of the switch are discussed in section 4.1. The switching policies implemented within this paper are given in section 4.2.

4.1 The Switching Process

Several different scenarios for server switching are presented in the literature [9, 14]. In [9] it is proposed that the set of servers are shared amongst a single application, which is partitioned according to different levels of QoS. In this case, the simplest approach to reallocating a server would be remove it from an entry point serving one request stream, and add it to the entry point for the assigned pool. This negates the need to undeploy and redeploy the application, which provides a considerable reduction in switching cost. The switching process for this scenario is given in algorithm 1.

There is a cost associated with switching a server from one application to another. The cost of a switch is derived from the duration of the switch, and can be considered as the degradation of the throughput in the environment whilst a server is unable to service requests for any application as it switches.

4.2 Switching Policies

A switching policy is defined as an algorithm that when provided information on the current state of the system makes a decision on moving to another state. When doing this the policy must analyze the potential improvement in QoS against the cost of performing the server switch. There are several examples of switching policies in the literature [9, 14]. Some of these policies are executed as a result of each arrival or departure of a request; while others are executed after

Algorithm 1 Switching algorithm for single application QoS requirements

```

1: for Application  $A_i$ , in applications  $A_{1..n}$  do
2:   Let  $S_i$  be servers required for  $A_i$ 
3:   Let  $AS_i$  be an application server belonging to  $A_i$ 
4:   Let  $W_i$  be a Web Server belonging to  $A_i$ 
5:   while  $S_i \neq 0$  do
6:     if  $S_i > 0$  then
7:       for  $A_m$  in  $A_{i+1..n}$  do
8:         if  $S_m < 0$  then
9:           Stop  $W_m$  dispatching requests to  $AS_i$ 
10:          Wait for pending requests to complete
11:          Switch server from  $A_m$  to  $A_i$ 
12:          Allow  $W_i$  to dispatch requests to  $AS_i$ 
13:        end if
14:      end for
15:    else
16:      for  $A_m$  in  $A_{i+1..n}$  do
17:        if  $S_m > 0$  then
18:          Stop  $W_i$  dispatching requests to  $AS_i$ 
19:          Wait for pending requests to complete
20:          Switch server from  $A_i$  to  $A_m$ 
21:          Allow  $W_m$  to dispatch requests to  $AS_i$ 
22:        end if
23:      end for
24:    end if
25:  end while
26: end for

```

a fixed time period and use statistics gathered over a time window to inform the switching decision. A policy may also consider request arrivals as being *on* or *off*, which is dictated by any arrivals in a given time period. The work presented in [14] describes four possible switching policies, three of which are implemented in this paper:

- The *Average Flow Heuristic* uses information on the arrival and completion rates of requests for each application in order to make a switching decision. This heuristic averages arrivals over the duration of the experiment and does not consider the distinct on/off periods for each application. Doing this requires that a weighted average arrival rate is calculated; this is shown in algorithm 2. Algorithm 4 is then used with the calculated average arrival rates.
- The *On/Off Heuristic* attempts to consider the “bursty” nature of requests to each application. To do this it classifies each application’s requests as being on or off, and switches servers accordingly. To account for the on and off periods in the job streams, the arrival rate is calculated as in algorithm 3; algorithm 4 is then used to calculate a new server allocation.
- The *Window Heuristic* uses statistics gathered over a sliding window of time to calculate arrival and completion rates for each application within

Algorithm 2 Calculating the reduced arrival rate for the Average Flow Heuristic

Input: Arrival rate λ
Job stream on rate m
Job stream off rate n
Output: Average arrival rate λ'
return $\frac{\lambda \times m}{m+n}$

Algorithm 3 Calculating the arrival rate for the On/Off Heuristic

Input: Arrival rate λ
Job stream on period m
Output: New Arrival rate λ'
if $m = true$ **then**
 return λ
else
 return 0
end if

a time window. In so doing, the policy ignores the presence of any off periods in the time window. This algorithm is shown in algorithm 6.

5 Experimental Platform

In this paper we present our investigations into the single application with multiple QoS requirements, as found in [9].

5.1 Experimental Platform

Our experimental platform is based on the architecture shown in figure 1. In the presentation tier we use a custom Web server to dispatch requests onto the application servers. The glassfish J2EE application server running on a Java 1.6 JVM was selected for the application runtime environment. The application server was tuned in accordance with the manufacturer's published guidelines to improve performance [15]. For the data persistence tier the Oracle 10g relational database system was chosen, which is representative of production systems that one might find in the field.

The hardware for the Web servers consists of two dual Intel Xeon 2.0GHz servers with 2GB of RAM. For the application servers, a server pool of eight homogeneous servers is used. The servers all use dual Intel Xeon 2.0 GHz processors and had 2GB RAM installed. They are connected via a 100 Mbps ethernet network. The web servers for each application were comprised of the same hardware. The database servers were all configured as dual Intel Xeon 3.0Ghz CPU servers with 2GB RAM and were connected to the network via a gigabit ethernet connection.

The application used for the testing of the system was Daytrader [2], an open-source version of IBM's performance benchmark Trade. This application was chosen as it is representative of a high throughput Web application. The

Algorithm 4 Server Allocation Algorithm**Input:** Current server allocation S_1, S_2

Arrival Rates, λ_1, λ_2
 Completion Rates, μ_1, μ_2
 Queue Lengths q_1, q_2
 Switches in progress $w_{1,2}, w_{2,1}$
 Switch Rate $r_{1,2}, r_{2,1}$
 Job costs c_1, c_2
 Switch costs $sc_{1,2}, sc_{2,1}$

Output: New server allocation, S'_1, S'_2

```

1: Let  $tc_1, tc_2$  be total costs for each job queue
2:  $tc_1, tc_2 \leftarrow 0$ 
3: Let  $bdc$  be best decision cost
4:  $bdc \leftarrow \infty$ 
5: if  $\mu_1 = 0$  and  $\mu_2 = 0$  then
6:   return error
7: end if
8: for  $s$  in  $S_1$  do
9:    $tc_1 \leftarrow$  Call Algorithm 5 with parameters  $s, S, \lambda_1, \mu_1, w_{2,1}, r_{2,1}, q_1$ 
10:   $tc_2 \leftarrow$  Call Algorithm 5 with parameters  $s, S, \lambda_2, \mu_2, w_{1,2}, r_{1,2}, q_2$ 
11:  if  $(c_1 \times tc_1 + c_2 \times tc_2 + sc_{1,2} \times s) < bdc$  then
12:     $S'_1 \leftarrow -s$ 
13:     $S'_2 \leftarrow s$ 
14:  end if
15: end for
16: for  $s$  in  $S_2$  do
17:    $tc_1 \leftarrow$  Call Algorithm 5 with parameters  $s, S, \lambda_1, \mu_1, w_{2,1}, r_{2,1}, q_1$ 
18:    $tc_2 \leftarrow$  Call Algorithm 5 with parameters  $s, S, \lambda_2, \mu_2, w_{1,2}, r_{1,2}, q_2$ 
19:   if  $(c_1 \times tc_1 + c_2 \times tc_2 + sc_{2,1} \times s) < bdc$  then
20:      $S'_1 \leftarrow s$ 
21:      $S'_2 \leftarrow -s$ 
22:   end if
23: end for
24: return  $S'_1, S'_2$ 

```

work presented in [1] suggests adopting an exponential distribution with a mean of seven seconds as a reasonable “think time” for the trade application.

To generate dynamic workloads a custom load generation system was developed. This allows specified load to be generated for predetermined durations, which allowed us to monitor the reaction of the switching system to repeatable changes in workload. All of the policies were subject to an identical workload. The workload consisted of one thousand client sessions, which were initially divided between the applications and were altered during the execution of the experiment. This allowed us to observe the reaction of each policy under a consistent environment. The workload is shown in table 1.

To host the switching system, an additional node was added to the architecture in figure 1. This was done to ensure that the additional overheads of the system were not added to any of the existing system components.

Algorithm 5 Total Cost Algorithm

Input: Switched servers s Server Allocation S Arrival rate λ Completion rate μ Switches in Progress $w_{m,n}$ Switch rate $r_{m,n}$ Queue Length q **Output:** Total Cost, tc

```

1: if  $q > 0$  then
2:   if  $\lambda < S - s + w_{m,n} \times \mu$  then
3:     Let  $st$  be an array of size  $w_{m,n} + 1$ 
4:     for  $i$  in  $w_{m,n}$  do
5:        $st_i \leftarrow \frac{1}{(w_{m,n} - i) \times r_{m,n}}$ 
6:     end for
7:      $st_{w_{m,n}} \leftarrow \infty$ 
8:      $tc_1 = 0$ 
9:     Let  $vq$  be the virtual queue length
10:     $vq \leftarrow q$ 
11:    for  $j$  in  $w_{m,n} + 1$  do
12:      if  $vq > 0$  then
13:        Let  $x$  be the rate at which the queue drains
14:         $x \leftarrow vq + (\lambda - (S - s + j) \times \mu) \times st_j$ 
15:        if  $x \geq 0$  then
16:           $tc \leftarrow tc + 0.5 \times (vq + x) \times st_j$ 
17:           $vq \leftarrow x$ 
18:        else
19:           $tc \leftarrow tc + 0.5 \times \frac{-vq}{\lambda - (S - s + j) \times \mu \times vq}$ 
20:           $vq \leftarrow 0$ 
21:        end if
22:      end if
23:    end for
24:  else
25:     $tc \leftarrow \infty$ 
26:  end if
27: else
28:    $tc \leftarrow 0$ 
29: end if
30: return  $tc$ 

```

Although the time taken to switch a server varies, and is in part dependent on the queue of pending requests allocated to the server, we have found that the average time taken to switch a server between pools is approximately 4 seconds¹.

The *switching interval* is the time between executions of the switching policy. In this experiment the switching interval selected was thirty seconds, as this

¹The range of switching times obtained throughout the experiments ranged from 2 to 6.4 seconds.

Algorithm 6 Window Policy Algorithm

Input: Current server allocation S_1, S_2 ;

Arrival Rates, λ_1, λ_2

Completion Rates, μ_1, μ_2

Job costs, c_1, c_2
Output: New server allocation, S'_1, S'_2

```

1: Let  $bdc$  be best decision cost
2:  $bdc \leftarrow \infty$ 
3:  $n_1 = \frac{(s_1+s_2) \times c_1}{c_1, c_2}$ 
4:  $n_2 = (s_1 + s_2) - n_1$ 
5: for  $i$  in  $S_1 + S_2$  do
6:    $\rho_1 = \frac{\lambda_1}{i \times \mu_1}$ 
7:    $\rho_2 = \frac{\lambda_2}{(S_1+S_2-i) \times \mu_2}$ 
8:   if  $\rho_1 < 1$  and  $\rho_2 < 1$  then
9:     Let  $c$  be cost of the switch
10:     $c = \frac{c_1 \times \rho_1}{1-\rho_1} + \frac{c_2 \times \rho_2}{1-\rho_2}$ 
11:    if  $c < bdc$  then
12:       $bdc \leftarrow c$ 
13:       $n_1 = i$ 
14:       $n_2 = (S_1 + S_2) - i$ 
15:    end if
16:  end if
17: end for
18:  $S'_1 = n_1 - S_1$ 
19:  $S'_2 = n_2 - S_2$ 
20: return  $S'_1, S'_2$ 

```

allowed a complete switch of all servers from one pool to the other, if such behavior was required by any of the switching policies.

In the experiments we configure the two applications with different costs to represent the differences in QoS requirements. The *job costs* for our experiments are considered to be the costs for holding a job. Such a definition allows a value to be attached to a queue of waiting jobs. For our experiments *a1* has a holding cost 25% higher than that of *a2*, making jobs for *a1* a higher priority than *a2* as they are more expensive to hold.

Table 1: Application workload for all policies

		Timestep							
		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
Duration (mins)		1	1	1	1	1	1	1	1
Clients	Application 1 (<i>a1</i>)	800	800	600	600	400	400	200	200
	Application 2 (<i>a2</i>)	200	200	400	400	600	600	800	800

5.2 Results

The overhead of the system is measured by calculating the maximum throughput of a single server directly, and then measuring the maximum throughput of the server requests that are forwarded from the Web server. We measure the throughput for each case at a variety of loads as shown in figure 2. It can be observed that the throughput for the system is significantly higher than that of the direct connections. The throughput curves for both connection types fit closely with the typical performance curves seen in [6].

The response time for the direct requests increases dramatically after 100 clients, while the response time for the redirected requests remains constant. The authors believe that this is due to connections between two fixed points (the Web server and the application server) being cached at the Web server, reducing startup costs for each connection.

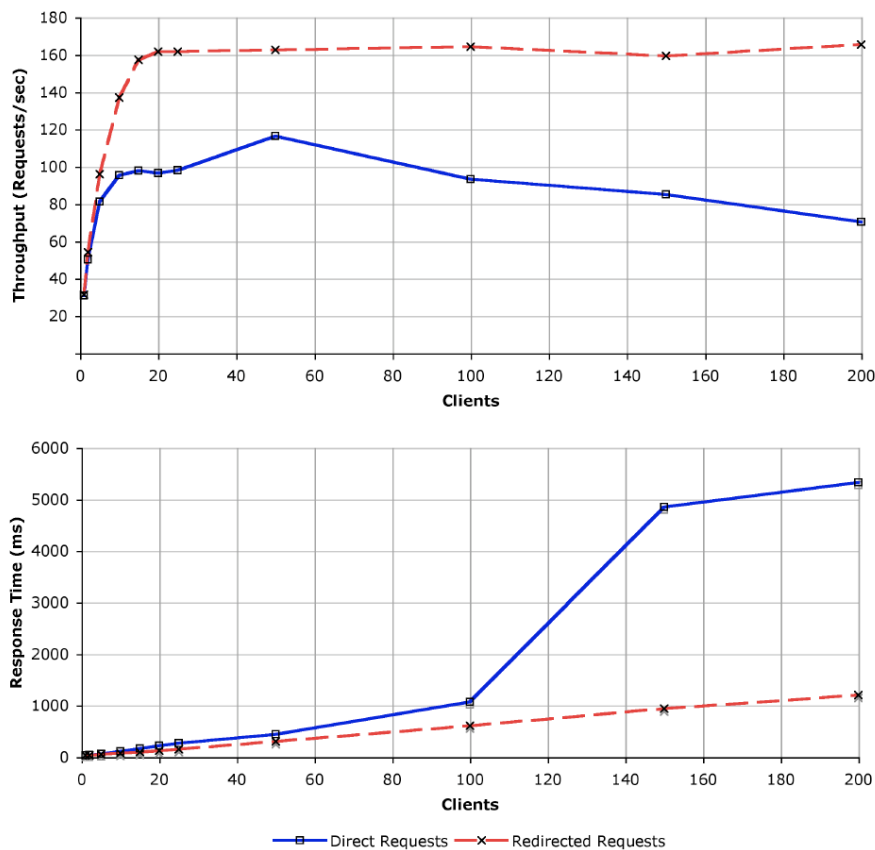


Figure 2: Direct Server Throughput vs. Redirected Throughput

A baseline is provided by statically allocating four servers to each of the two applications. The response time for the application workload shown in table 1, over a period of eight minutes, is shown in figure 3. The response times for each application at timestep 1 is high due to the application being freshly deployed.

After deployment the application server and the JVM optimise the the program to improve its performance. In the experiments, we have intentionally included the warm-up time as when a server is switched it will be starting from a cold state, and will take time to enter its steady state.

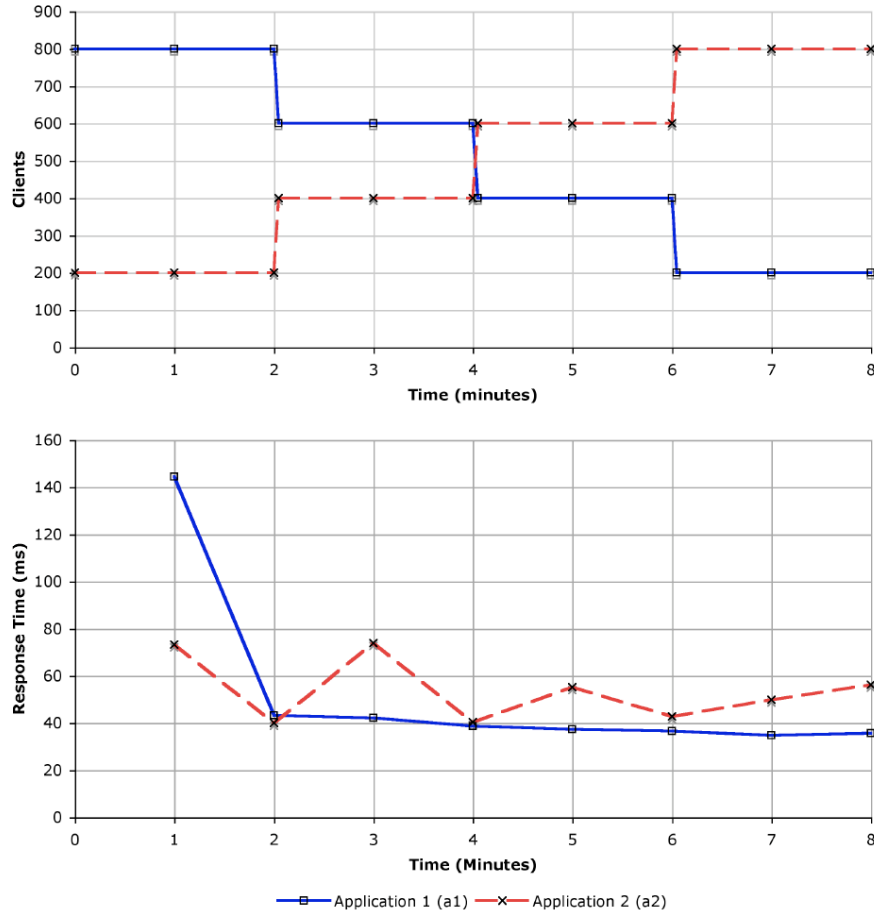


Figure 3: Response Times for Static Server Allocation

It can be seen in figure 3 that the response times for *a1* decrease after the first time period, and continue to decrease due to the decreasing workload placed upon the application throughout the experiment (see table 1). The response times for *a2* are similar, however they are more erratic throughout the experiment. Initially the response time for the application is lower than that of *a1* due to the reduced load. After two minutes the load on *a2* is doubled, which causes an increase in response time until the third minute. At this point, the response times are again reduced, which is due to the optimizations made upon the application by the JVM, in addition to the application server dynamically adjusting its internal configuration. Examples of this dynamic application server configuration include increasing the number of threads available in the database

Table 2: Comparison of policy response time against static allocation

	Mean response time (ms)			
	Static	Average Flow	On/Off	Window
<i>a1</i>	51.58	48.67 (-5.64%)	53.13 (3.01%)	54.25 (5.18%)
<i>a2</i>	53.92	52.03 (-3.51%)	52.46 (-2.71%)	50.02 (-6.31%)

Table 3: Comparison of policy throughput against static allocation

	Mean throughput (requests/second)			
	Static	Average Flow	On/Off	Window
<i>a1</i>	76.55	76.19 (-0.47%)	76.26 (-0.38%)	75.79 (-0.99%)
<i>a2</i>	76.49	75.75 (-0.97%)	76.42 (-0.09%)	76.22 (-0.35%)

connection pool during periods of high load. This behavior is exhibited between the fifth and sixth minutes in accordance with the changes in workload at this time.

After finding a baseline from the static server allocation, each of the three policies were measured against the same workload. The results of the three policies are shown in figures 4,5 and 6. The figures are set out as follows: the top graph represents the workloads for each application. The middle graph shows the server allocation throughout the experiment and the bottom graph displays the response time for each of the applications. The graphs are aligned such that the time represented on the x -axis is the same on all three graphs.

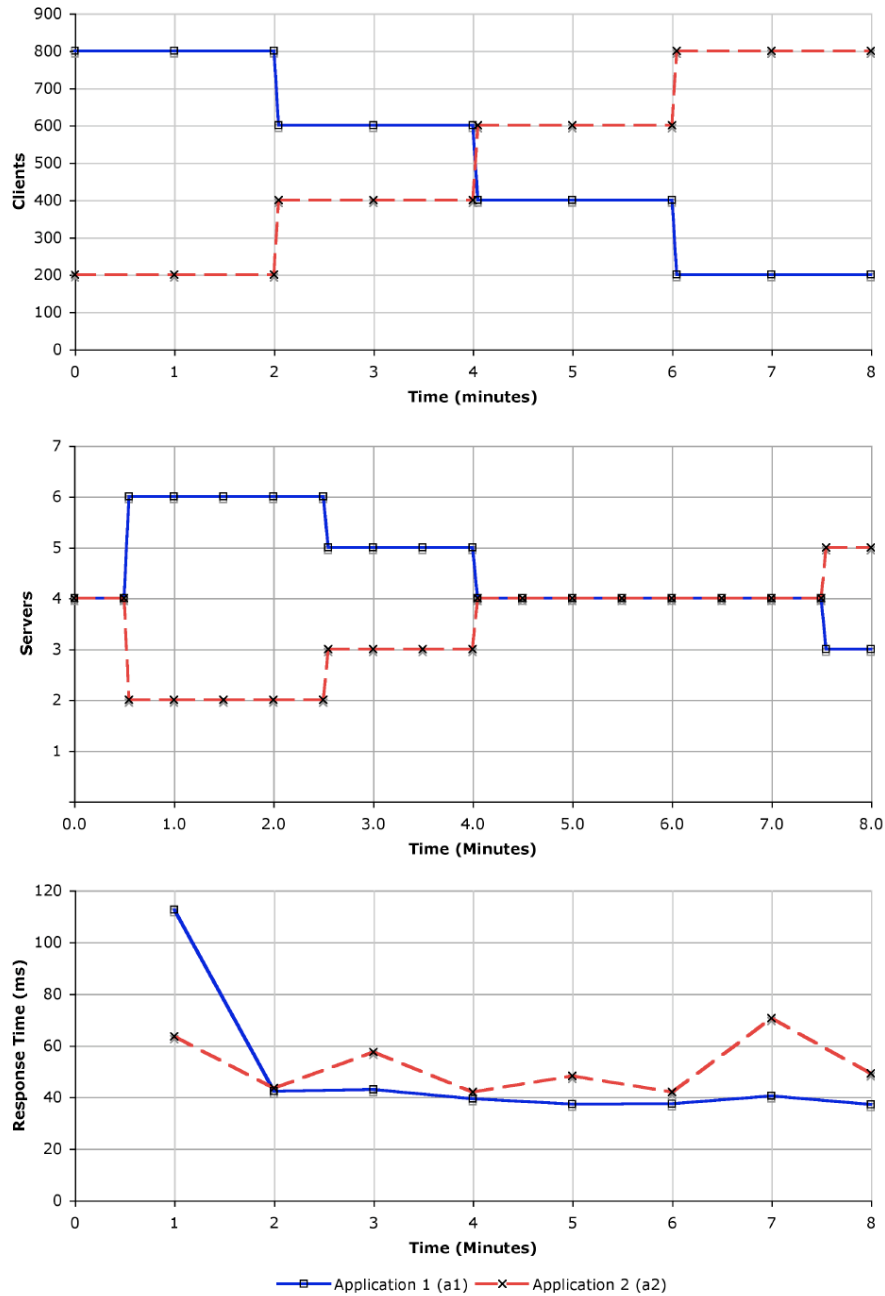
The performance of the Average Flow policy is shown in figure 4. When compared to the static allocation (see tables 2 and 3) the policy gives a 5.64% improvement in response time for *a1* and a 3.51% improvement for *a2*. The Average Flow Policy is the only policy to give an improvement for both applications. In terms of mean throughput, the Average Flow policy decreased throughput for *a1* by 0.47% and *a2* by 0.97%. The Average Flow policy performs the fewest server switches of any of the policies. This has the effect of making it the cheapest policy if we consider the cost of the switches as defined in section 4.

The On/Off policy results are shown in figure 5. The On/Off policy increases the response time for *a1* by 3.01%, but it reduces the response time for *a2* by 2.71%. The performance of this policy is therefore worse than the Average Flow policy, as it increases the response time for *a1*, and decreases the response time for *a2* by a smaller amount. When compared to the Average Flow policy, this policy improves the response times for *a2* in the last two minutes of the experiment as it switches a server sixty seconds earlier.

The Window policy (see figure 6) switched significantly more than the two previous policies. The response times for the Window policy show both the biggest increase for *a1* and the largest decrease for *a2*. The policy returned to an equal allocation of servers far more often than the other policies used here.

In terms of response time, all of the policies reduced the mean response time for *a2*, which was subject to increasing load throughout the experiment. The

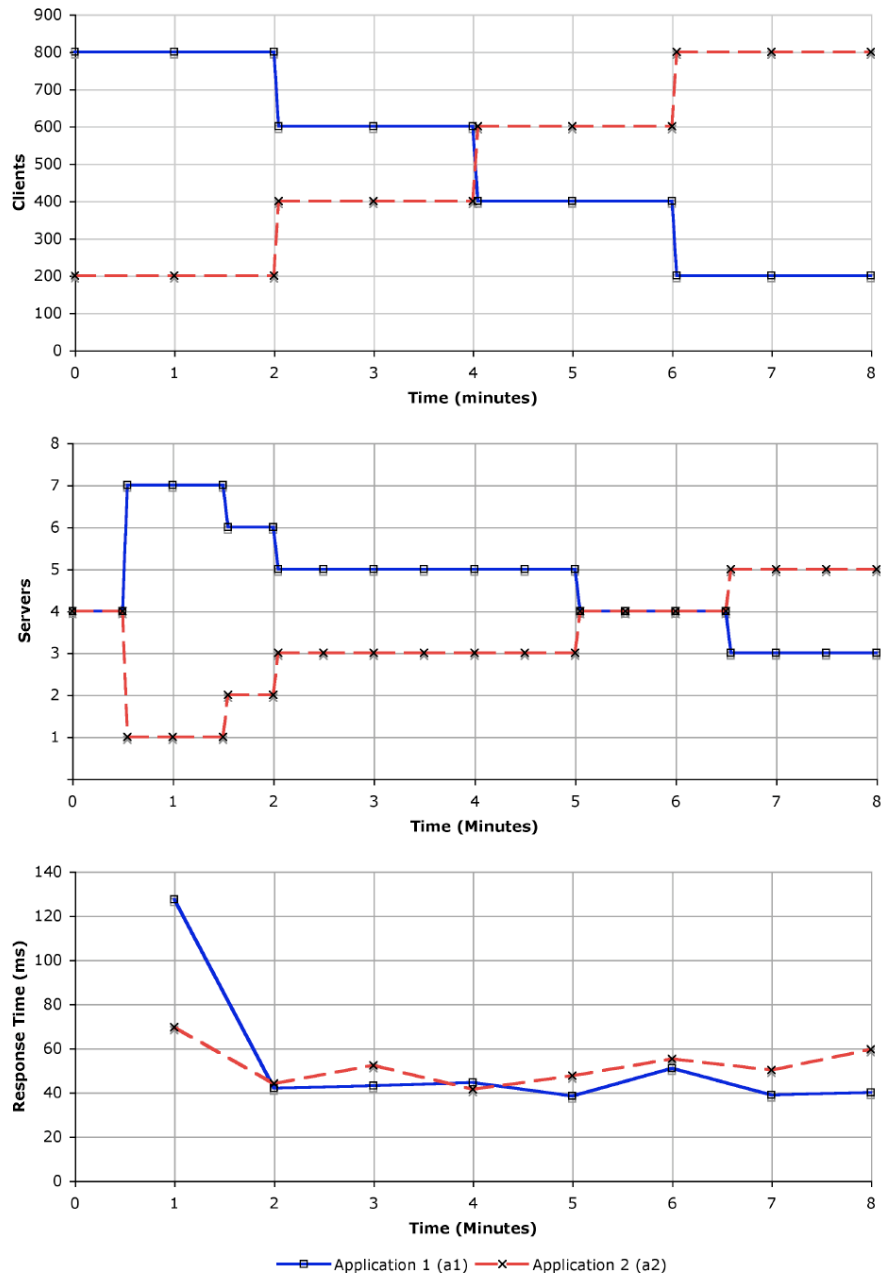
Figure 4: Average Flow Policy Results



mean response times for *a1* were increased by the On/Off and Window policies, however the Average Flow policy decreased the mean response time.

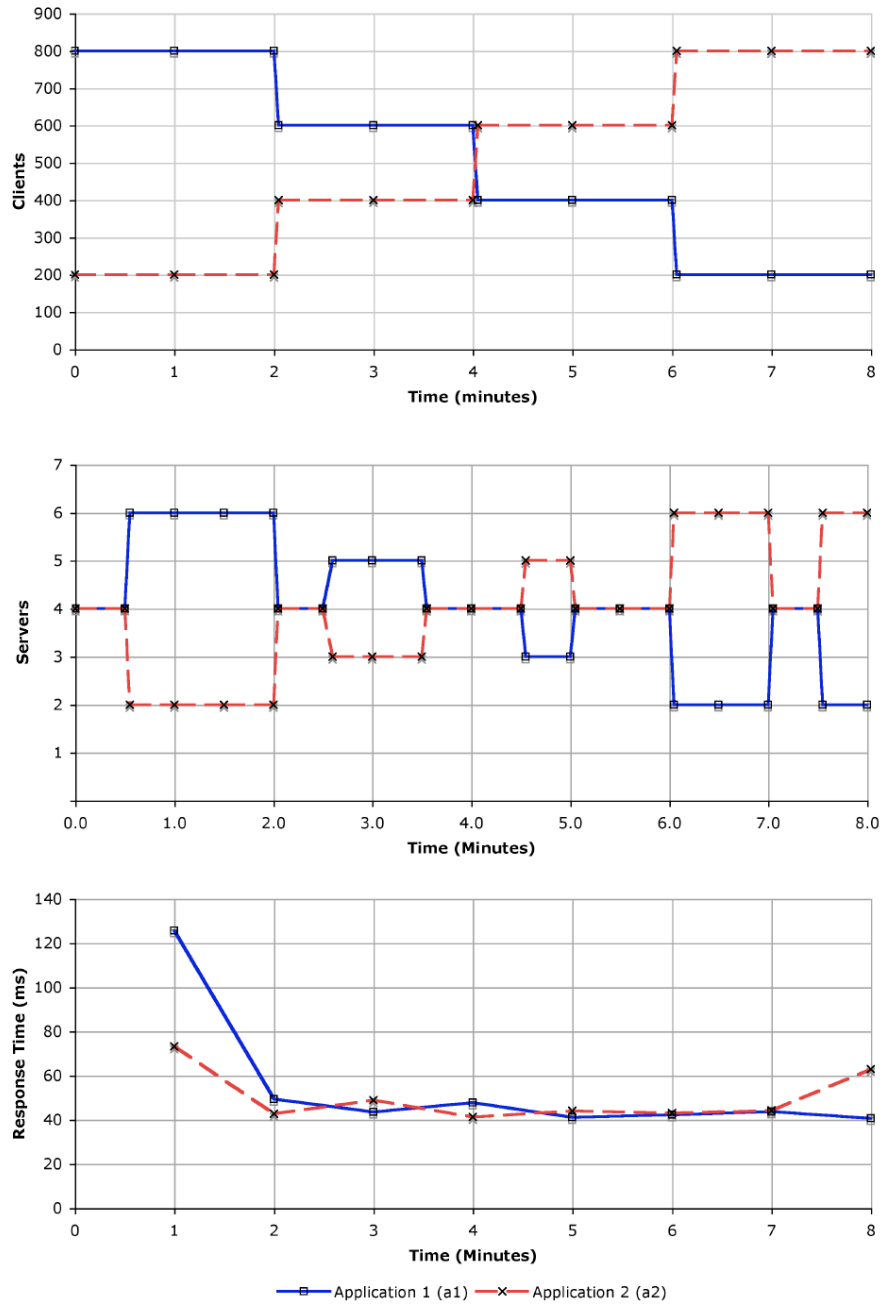
When considering the mean throughput of each policy against the static allocation it is unsurprising that the throughputs are lower as servers are unable

Figure 5: On/Off Policy Results



to process requests while they are being switched between pools. This inability to service requests is considered as the cost of making a switch as stated in section 4. It is worth noting however that in all cases the decrease in throughput

Figure 6: Window Policy Results



is less than 1%. In a system where changes in load are less frequent or switching intervals are longer, server switching would occur less frequently. In this case the effects of switching on throughput would be reduced further.

If we consider the mean response time as our metric for QoS, the Average

Table 4: Total switches performed by each policy

Policy	Timestep								Total Switches
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	
Average Flow Policy	2	0	1	0	1	0	0	1	5
On/Off Policy	3	1	1	0	0	1	1	0	7
Window Policy	2	0	3	1	1	1	2	4	14

Flow policy presents itself as the best policy for our given application. It was the only policy that reduced service times for both applications. In terms of the number of server switches (see table 4) the Average Flow performed five server switches; the On/Off policy performed seven server switches, and the Window policy performed fourteen over the course of the experiment. When considering the switching cost in terms of lost throughput, we can see that the Average Flow policy is however the most expensive, with the On/Off policy reducing throughput the least during the experiment.

6 Conclusion and Further Work

In this paper we have developed a switching system that is representative of a real world commercial hosting environment. We have implemented three theoretically derived switching policies as found in [14]. After implementing the three policies we have evaluated their respective performance within our testbed and identified the best policy for our specific application given a specific example workload.

There is a significant amount of research to be done in this area. In our experiments we used a fixed switching interval of thirty seconds which was appropriate for our experiments. This figure was derived through consideration of the switching duration of the specific application that we used for our experiments. Further investigations will focus on analyzing the switching duration.

The results for each policy shown here are derived from a fixed workload. In the short term we plan to investigate a variety of workload patterns, and identify policies which are most effective under specific workloads. The switching interval (which was fixed for our experiments) will be analyzed in conjunction with the workload patterns to consider its overall effect on the system.

In the longer term we will look to enhance the switching system by identifying known workload patterns in an application's workload trace and selecting the most appropriate policy from our experimentation. The identification of the workload patterns is also expected to determine the appropriate dynamic switching interval, allowing the most effective switching interval to be selected given known or predicted workload patterns.

Acknowledgments

This work is supported in part by the UK Engineering and Physical Science Research Council (EPSRC) contract number EP/C538277/1.

References

- [1] Y. An, T. Kin, T. Lau, and P. Shum. A scalability study for websphere application server and db2 universal database. Technical report, IBM, 2002.
- [2] The Apache Software Foundation, <http://cwiki.apache.org/GMOxDOC12/daytrader.html>. *Daytrader*.
- [3] D. A. Bacigalupo, J. W. J. Xue, S. D. Hammond, S. A. Jarvis, D. N. Dillenberger, and G. R. Nudd. Predicting the effect on performance of container-managed persistence in a distributed enterprise application. In *proc. 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS'07)*. IEEE Computer Society Press, March 2007.
- [4] S. A. Banawan and N. M. Zeidat. A Comparative Study of Load Sharing in Heterogeneous Multicomputer Systems. In *the 25th Annual Simulation Symposium*, pages 22–31, 1992.
- [5] L. Cherkasova and P. Phaal. Session Based Admission Control: a Mechanism for Peak Load Management of Commercial Web Sites. *IEEE Transactions on Computers*, 51(6), jan 2002.
- [6] G. Cuomo. A methodology for performance tuning. Technical report, IBM, 2000.
- [7] P. P. D. Villela and D. Rubenstein. Provisioning servers in the application tier for e-commerce systems. *ACM Transaction on Internet Technology*, 7(1):7, 2007.
- [8] K. Dutta, A. Datta, D. VanderMeer, H. Thomas, and K. Ramamritham. ReDAL: An Efficient and Practical Request Distribution Technique for Application Server Clusters. *IEEE transactions on Parallel and Distributed Systems*, 18(11):1516–1527, 2007.
- [9] L. He, W. J. Xue, and S. A. Jarvis. Partition-based Profit Optimisation for Multi-class Requests in Clusters of Servers. In *the IEEE International Conference on e-Business Engineering*, 2007.
- [10] Z. Liu, M. Squillante, and J. Wolf. On Maximizing Service-level-agreement Profits. *ACM SIGMETRICS Performance Evaluation*, 29:43–44, 01.
- [11] D. Menasce, V. A. F. Almeida, and et al. Business-oriented Resource Management Policies for e-Commerce Servers. *Performance Evaluation*, 42:223–239, 2000.
- [12] J. Palmer and I. Mitrani. Optimal and heuristic policies for dynamic server allocation. *Journal of Parallel and Distributed Computing*, 65(10):1204–1211, 2005.
- [13] N. G. ShiVaratri, P. Krueger, and M. Shingal. Load Distribution for Locally Distributed Systems. *IEEE Computer*, 8(12):33–44, December 1992.
- [14] J. Slegers, I. Mitriani, and N. Thomas. Evaluating the optimal server allocation policy for clusters with on/off sources. To appear.
- [15] Sun Microsystems, Inc. *Sun Java System Application Server 9.1 Performance Tuning Guide*, 2007.
- [16] K. Ueno, T. Alcott, J. Blight, J. Dekelver, D. Julin, C. Pfannkuch, and T. Sheih. Websphere scalability: Wlm and clustering, using websphere application server advanced edition (ibm redbook). Technical report, IBM, September 2000.
- [17] B. Urgaonkar, P. Shenoy, A. Chandra, and et al. Dynamic Provisioning of Multi-tier Internet Applications. In *Second International Conference on Autonomic Computing (ICAC'05)*, 2005.
- [18] M. Welsh and D. Culler. Adaptive Overload Control for Busy Internet Servers. In *the 2003 USENIX Symposium on Internet Technologies and Systems*, 2003.

The role of client behaviour in the performance analysis of eCommerce systems

D.R.W. Holton* I.U. Awan† M. Younas‡

Abstract

In order to investigate the performance of priority protocols in an eCommerce system, we simulate their behaviour when composed with clients with differing behaviour. As suspected, we find that there is a relationship between the behaviour of the clients and the performance of the protocols.

1 Introduction

We have been investigating a number of protocols for scheduling requests in eCommerce applications [6, 5], evaluating their performance in the context of a population of simulated clients with very simple behaviour. While we have been able to obtain useful results, we conjecture that providing the clients with more complex behaviour will give us deeper insight into the applicability of our protocols.

The system under investigation is composed of a population of p clients and a single eCommerce server – a π -calculus [11] description is given below in **Specification 1**:

Specification 1 Configuration of the eCommerce system

$$\begin{aligned} & \text{new service, } \overrightarrow{\text{reply}} \\ & \text{Client}(\text{service}, \text{reply}_1) \mid \text{Client}(\text{service}, \text{reply}_2) \mid \dots \\ & \mid \text{Client}(\text{service}, \text{reply}_p) \mid \text{Server}(\text{service}) \end{aligned}$$

We have examined the performance of this system with alternative (observationally equivalent) servers and now turn our attention to the clients, with the intention of investigating how factors such as response time and rejected requests affect performance. We first introduce our approach for defining client behaviour and illustrate it by describing two different clients, then describe the implementation. Performance results are then presented and analysed. Finally some conclusions are drawn and further work is suggested.

*Department of Computing, The University of Bradford, d.r.w.holton@comp.brad.ac.uk

†Department of Computing, The University of Bradford, i.u.awan@bradford.ac.uk

‡Department of Computing, Oxford Brookes University, m.younas@brookes.ac.uk

2 Defining client behaviour

The approach is based on Client Behaviour Modeling Graphs (CBMG) [10], augmented to allow a client's behaviour to depend on responses received from a server. CBMGs are recovered from analysis of web server logs and give a very clear indication of the **observable** behaviour of clients. However, in order to improve the analysis of our protocols for prioritising queries in eCommerce applications, we recognised that it was necessary to define the **unobservable** behaviour of clients, particularly clients' responses to the behaviour of the server. We have chosen to focus on the behaviour of clients when there are delays in receiving responses to queries and when queries are rejected, though the approach is completely general and may be used to account for any other behaviour such as clients' response to following dead links. In effect, we are turning the "spontaneous" exit actions from [10] into responses to the behaviour of the eCommerce application.

To illustrate this, figure 1(a) shows a client which generates buy and browse requests and exits at some point, while figure 1(b) explains **why** the client behaves in this way by showing that the client responds to receiving success messages by generating new requests, while it might exit after receiving a reject message or a timeout occurs.

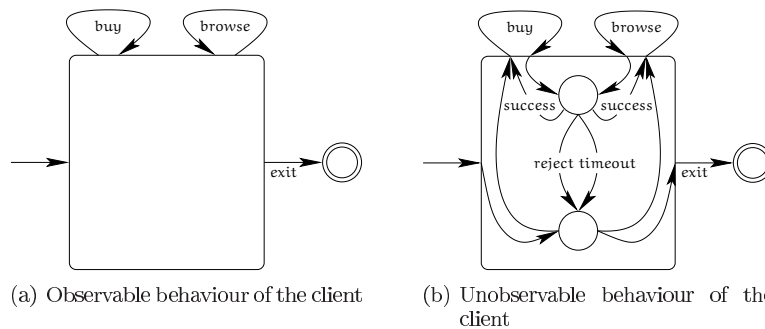


Figure 1: Illustrating observable and unobservable client behaviour

We chose to use the π -calculus instead of a graphical notation because complex behaviour can be expressed compactly, and its compositional nature allows the substitution of one component for another – both of these attributes will be illustrated below. In addition, the ability to pass actions as parameters allows truly elegant models to be constructed. We chose not to use a stochastic process algebra because the complexity of the system meant that we intended to construct a simulation rather than attempt to find analytical solutions, thus the extra (though admittedly small) overhead of including rates in the model was felt to be unnecessary.

The preliminary investigations of our scheduling protocols made use of a client that generated five requests, selecting buy requests with a probability of 0.2. No account was taken of the performance of the server – the client waited patiently for its request to be completed, and if its request was rejected due to server overload it did not leave the site in frustration. This is not typical behaviour for users of eCommerce systems, for whom instant gratification is too

slow.

To overcome the limitations in the client described above, we defined a simple client (**Specification 2**) which generates k requests and reacts to excessively long response times or rejected requests by either exiting or submitting a new request. Note how compactly this behaviour has been captured, and how easy it was to parameterise the specification with k . The ability to pass actions as parameters allows the actual value of the timeout action in Client_p^i to depend on the type of request attempted. The non-deterministic choice operator (+) allows alternative behaviours to be composed without specifying how the choice between those alternatives is made. For example, the choice in Client_a^n might be implemented as a probabilistic choice [1], while the choice between reply and timeout in Client_p^i might be considered a race condition. This allows us to investigate how the components of the system interact while permitting us flexibility when implementing the specification.

Specification 2 The simple client

$$\begin{aligned}
& \text{Client}_a^n(\text{request}, \text{reply}) \stackrel{\text{def}}{=} \overline{\text{request}}(\text{browse}, \text{reply}). & 1 \leq n \leq k \\
& \quad \text{Client}_p^n(\text{request}, \text{reply}, \text{timeout}_{\text{browse}}) & k \in \mathbb{N}_1 \\
& + \overline{\text{request}}(\text{buy}, \text{reply}).\text{Client}_p^n(\text{request}, \text{reply}, \text{timeout}_{\text{buy}}) \\
& \text{Client}_a^{k+1}(\text{request}, \text{reply}) \stackrel{\text{def}}{=} \mathbf{0} \\
& \text{Client}_p^i(\text{request}, \text{reply}, \text{timeout}) \stackrel{\text{def}}{=} & i \in \mathbb{N}_1 \\
& \quad \text{reply}(\text{result}). \\
& \quad (\text{if } \text{result} = \text{success} \text{ then } \text{Client}_a^{i+1}(\text{request}, \text{reply}) \\
& \quad + \text{if } \text{result} = \text{reject} \text{ then } \text{Respond}_r^i(\text{request}, \text{reply})) \\
& + \text{timeout}.\text{Respond}_t^i(\text{request}, \text{reply}) \\
& \text{Respond}_t^i(\text{request}, \text{reply}) \stackrel{\text{def}}{=} \text{exit}.\mathbf{0} + \text{Client}_a^{i+1}(\text{request}, \text{reply}) \\
& \text{Respond}_r^i(\text{request}, \text{reply}) \stackrel{\text{def}}{=} \text{exit}.\mathbf{0} + \text{Client}_a^{i+1}(\text{request}, \text{reply})
\end{aligned}$$

We then defined a client which attempted to perform a transaction comprising $k - 1$ browse requests followed by a single buy request, reacting to the server's responses in the same way as the simple client. As can be seen in **Specification 3**, this was achieved by giving a new definition for Client_a ; the other components of the simple client were reused.

The performance of **Specification 1** has been investigated when a population of either simple clients or transactional clients is composed with the eCommerce server. Results from those investigations are presented below.

Specification 3 The transactional client

$$\begin{aligned}
 \text{Client}_a^n(\text{request}, \text{reply}) &\stackrel{\text{def}}{=} \overline{\text{request}}(\text{browse}, \text{reply}). \\
 &\quad \text{Client}_p^n(\text{request}, \text{reply}, \text{timeout}_{\text{browse}}) \\
 \text{Client}_a^k(\text{request}, \text{reply}) &\stackrel{\text{def}}{=} \overline{\text{request}}(\text{buy}, \text{reply}). \text{Client}_p^k(\text{request}, \text{reply}, \text{timeout}_{\text{buy}}) \\
 \text{Client}_a^{k+1}(\text{request}, \text{reply}) &\stackrel{\text{def}}{=} \mathbf{0}
 \end{aligned}
 \quad \begin{array}{l} 1 \leq n < k \\ k \in \mathbb{N}_1 \end{array}$$

3 Performance results and analysis

3.1 Description of the implementation

We have constructed a multi-agent simulation [3] framework in Ada to evaluate the performance of our scheduling protocols. Ada was designed to support concurrent programming, enabling a close match between the π -calculus specifications and their implementation. The scheduling protocols were implemented as task types and since the signatures of those task types are all the same, it is possible to substitute one protocol for another in the framework by simply changing a (one-line) context clause. The same approach was taken to implementing the clients described above. Ada's usual mechanism for inter-process communication, the rendezvous, was not used since it does not match actions in the π -calculus. Instead we used the implementation of actions described in [4].

Our prioritising protocols are based on the assumption that buy requests should have a higher priority than browse requests, since the aim of eCommerce systems is to sell things. For all protocols, requests are sorted into separate buy and browse queues, with the difference between the protocols being the dequeuing mechanism. Once a queue is full the server rejects requests of that type until space is made available by processing queued requests. For the purposes of this comparison we selected our **fair scheduler** and our **threshold scheduler**, described below (the π -calculus descriptions are not given since the focus of this paper is the client behaviour).

The fair scheduler is based on multi-level queueing [13] with the following modifications:

1. if b consecutive buy requests have not been processed, then the first buy request is processed;
2. if b consecutive buy requests have been processed, then process (up to) the first c browse requests.

Parameters b and c allow the relative priorities of buy and browse requests to be adjusted.

The threshold scheduler uses a threshold for each queue, T_{buy} and T_{browse} , and works as follows:

1. if the length of the buy queue is greater than or equal to T_{buy} , or the length of the browse queue is less than T_{browse} , then process the first buy request;

2. if the length of the browse queue is greater than or equal to T_{browse} , and the length of the buy queue is less than T_{buy} , then process the first browse request.

This behaviour allows the scheduler to react to numerous lower priority browse requests arriving in a burst, while still giving priority to buy requests.

If there are no queued buy requests, both schedulers allow browse requests to be processed freely. Similarly, if there are no queued browse requests, neither scheduler waits for one to become available since this could lead to priority inversion [12].

The non-deterministic choices in the clients defined above were implemented as follows:

- choices where the outcome was determined by probabilities (e.g. choice between exiting or proceeding to the next request) were implemented by generating a random number and testing its value in an **if ... then ... else** statement;
- choices where the outcome was determined by a value (e.g. the server indicating that a request was successful) were implemented by testing that value in an **if ... then ... else** statement;
- choices where the outcome was determined by a race condition (e.g. whether the client receives a reply before it times out) were implemented using Ada's asynchronous transfer of control (ATC) statement.

These alternatives sufficed to implement the clients defined above, but finding more general ways of implementing non-deterministic choice is a difficult problem.

3.2 Performance results

The results presented below were obtained with a population of 2000 clients, with mean arrival rates varied as shown. The server processed requests with a mean rate of 40 per second. The probability of the simple client generating a buy request was 0.2. For both clients, the number of requests generated (k) was five, the timeout for a browse request was taken to be one second, while the timeout for a buy request was three seconds. The probability of exiting after a timeout was 0.05, and the probability of exiting after a rejected request was 0.15. The buy and browse buffers both had eight places. For the fair server $b = 2$ and $c = 1$, meaning that twice as many buy requests will be selected as browse requests. For the threshold server, $T_{\text{buy}} = T_{\text{browse}} = 4$.

Throughputs of the protocols for either client are shown in figure 2(a). These are as expected, and indicate that neither protocol imposes an excessive run-time overhead since the throughputs achieved are close to the mean of 40 specified above.

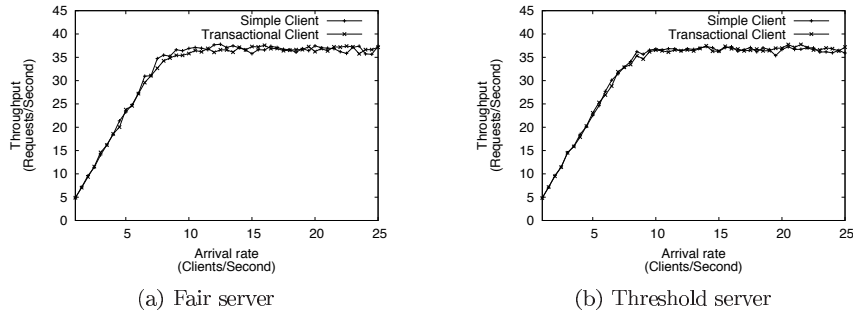


Figure 2: Comparison of throughputs

Figure 3(b) shows that the threshold server does not reject buy requests from either client type indicating that, under heavy load conditions, it manages to process all its higher priority requests at the expense of rejecting increasing numbers of lower priority requests. However, figure 3(a) shows that the fair server rejects up to 10% of buy requests from the simple client, but rejects none from the transactional client. The fact that buy requests are rejected is to be expected from the definition of the fair client, especially given the choices for b and c above.

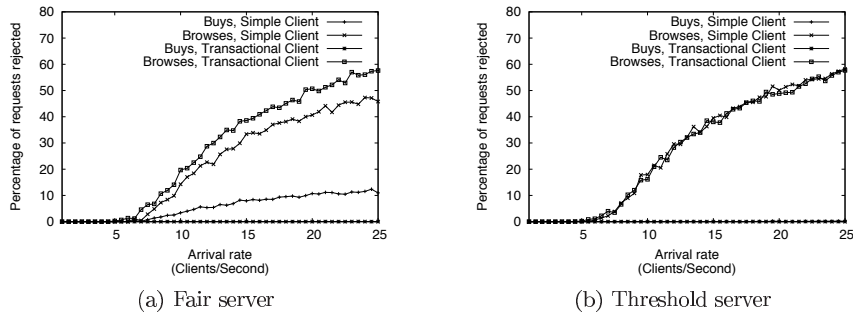


Figure 3: Comparison of rejected requests

Figure 4(a) is the key to understanding why no buy requests issued by the transactional client are rejected: considerably fewer buy requests are generated. This is explained by the fact that the buy request comes at the end of the transaction, so any of the four previous low priority requests may fail or timeout, potentially causing the client to exit. A natural question to ask from this result is whether inverting the priority of buy and browse requests might allow more transactional clients to complete their transaction? This question is answered affirmatively in figure 4(b), where results are obtained from setting $b = 1$ and $c = 2$, i.e. after processing one buy request, process two browse requests.

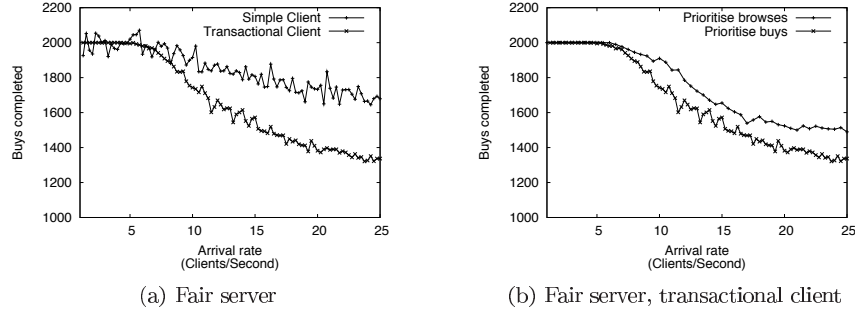


Figure 4: Comparison of Buy requests completed

4 Conclusions

Instead of making the effort of constructing a complex simulation, why not replace our population of clients with a single, equivalent, traffic-generating component? If we define a “traffic simulation” relation, \sim^T , then perhaps we could find a single component to replace the population of clients,

$$\text{TrafficGenerator}(\text{service}, \overrightarrow{\text{reply}}) \sim^T \text{Client}(\text{service}, \text{reply}_1) \mid \dots \mid \text{Client}(\text{service}, \text{reply}_p)$$

and compose this with the server instead,

$$\text{TrafficGenerator}(\text{service}, \overrightarrow{\text{reply}}) \mid \text{Server}(\text{service})$$

with the result of having a simpler system to analyse. Unfortunately, the results obtained demonstrate that the exact nature of the traffic generated depends on the characteristics of the server, and that the performance of our scheduling protocols is an emergent property of the interaction between the clients and the server. Seemingly small changes in the behaviour of either or both entities leads to large changes in performance. While this is not an original approach, see e.g. [7, 9, 2], little appears to have been published describing performance results obtained in this way. The majority of work done in this area is targeted towards analysis of the observed behaviour of clients, e.g. [14, 10, 8]. A reasoned analysis of why user emulation is difficult is given in [7], though we disagree with their conclusions. One objection is that the relationship between aggregate properties of traffic (e.g. self-similarity) and the behaviour of individual clients is hard to establish. We would argue that such properties would emerge as a result of the interaction between the clients and the server, though further development and analysis of our approach would be required to demonstrate this. Another objection is that accounting for network traffic between clients and server is important, and effects such as variable network delays and retransmission should be built into the client behaviour. We would argue against this on the grounds of separation of concerns: a separate component should be responsible for directing the traffic between server and clients, with the added benefit that different network types (LAN, WiFi) can be modelled and their effect upon performance evaluated. The third concern, scalability, is taken care of by Moore’s Law.

Whilst a variety of other methods may be used for describing client behaviour, we have found the π -calculus to be very effective, as well as a useful starting point for the implementation.

We intend to develop this work in a number of ways, including:

- discover more realistic values for simulation parameters such as timeouts and exit probabilities;
- examine the effect of probability distributions other than exponential on the performance results;
- examine the traffic generated by the interaction between clients and server to see what aggregate properties it exhibits;
- evaluate our scheduling protocols and see if it is possible to match protocols to traffic types.

References

- [1] S. Andova. Process Algebra with Probabilistic Choice. *Proceedings of the 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems*, pages 111–129, 1999.
- [2] M. Deshpande and G. Karypis. Selective Markov models for predicting Web page accesses. *ACM Transactions on Internet Technology (TOIT)*, 4(2):163–184, 2004.
- [3] J. Ferber. *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley, Harlow, 1999.
- [4] D.R.W. Holton. Implementing π -calculus style actions. Technical report, University of Bradford, 2008. <http://www.comp.brad.ac.uk/~drwholto/implementing-pi.pdf>.
- [5] D.R.W. Holton, I. Nafea, M. Younas, and I. Awan. A Class-based Scheme for E-Commerce Web Servers: Formal specification and Performance Evaluation. Accepted by the Journal of Network and Computer Applications, Elsevier, 2008.
- [6] D.R.W. Holton, M. Younas, and I. Awan. Priority Scheduling of Requests to Web Portals. Submitted to DEXA '08, 2008.
- [7] K. Kant, V. Tewari, and R. Iyer. Geist: a generator for e-commerce & internet server traffic. *Performance Analysis of Systems and Software, 2001. ISPASS. 2001 IEEE International Symposium on*, pages 49–56, 2001.
- [8] D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira Jr. In search of invariants for e-business workloads. *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 56–65, 2000.
- [9] D.A. Menasce. Trade offs in designing web clusters. *IEEE Internet Computing*, September/ October 2002.

- [10] D.A. Menasce, V.A.F. Almeida, R. Fonseca, and M.A. Mendes. A methodology for workload characterization of e-commerce sites. In *ACM Conference on Electronic commerce*, pages 119–128, 1999.
- [11] R. Milner. *Communicating and Mobile systems: the π -Calculus*. Cambridge University Press, 1999.
- [12] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, 1990.
- [13] A. Silberschatz, P.B. Galvin, and G. Gagne. *Operating System Concepts*. Wiley, New York, 2004.
- [14] O.R. Zaiane, M. Xin, and J. Han. Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs. *Research and Technology Advances in Digital Libraries, 1998. ADL 98. Proceedings. IEEE International Forum on*, pages 19–29, 1998.

Fine Grain Stochastic Modeling and Analysis of Low Power Portable Devices with Dynamic Power Management

Yuan Chen, Fei Xia, Delong Shang, Alex Yakovlev

Asynchronous Systems Laboratory

Newcastle University

{Yuan.chen1, Fei.xia, Delong.shang, Alex.yakovlev}@ncl.ac.uk

Abstract

Dynamic Power Management (DPM) is one of the main system-level methods for power-aware design of portable devices. Different DPM policies can be used to give on-off control to computation units in portable devices that are driven by external events such as incoming data or harvested energy. With power control becoming more complex and accurate, the power and latency cost for the power management unit increases as well. This paper presents a method of constructing stochastic models for DPM systems when the Power Control unit is not taken as cost free. These models can be used for fine grain analysis of system performance.

1. Introduction

With rapid progress in semiconductor technology, portable devices are enabled with sophisticated processing capability and can provide services that were only available in desktop computers decades ago. However, the high frequency and chip density in new designs not only brings fast execution performance, but also makes the battery-operated system more energy hungry. Therefore, low-power design which tries to reduce the power dissipation while still satisfying the performance requirement becomes a hot research field in electronic research.

System level energy-saving technologies focus on increasing the energy efficiency in portable devices. Dynamic Power Management (DPM) [1], for example, will provide power on-off control to the portable device whose computation units are event-driven for reactive processing. These units are activated and can access the battery power only when they are triggered by some external events to carry out the corresponding tasks and a so-called power management unit will be added to the system where some scheme (policy) is implemented to decide when and how to shutdown or wakeup certain units. Different from other computation units in the system, the power management unit, serving as an event watchdog, will always be active.

In various portable devices, the concept of events may have different meanings. It may represent for example incoming data requiring execution or communication requests for routing calculation. At a high level of abstraction, they may be understood as events with the capability of causing task execution. For implementations based on energy harvesting [13], events can also be the availability of energy that can be collected, which enables and thus causes tasks to be executed. Such “energy tokens” or “energy events” have the same practical effect at a high level of modeling to the more traditional data and signal events. This harvestable energy

highly depends on the environment and is usually not enough to keep computation units constantly active, thus making a power management scheme indispensable.

According to the survey by Benini et al. in [2], DPM policies can be generally grouped into predictive or stochastic policies. *Prediction* policies make shutdown/wakeup decisions according to the processing of history data of previous incoming events, and the efficiency of these policies greatly relies on the amount and correlation of history data used for calculation. *Stochastic* policies, on the other hand, consider the nondeterminism of the workload as well as the processing and transitions of the system, and mostly use Markov models for analysis. The applicability of modeling portable devices as Markov processes has been thoroughly analyzed in [4]. Discrete-time [2] and Continuous-time Markov chains [3] have been used for analyzing system-level power reduction techniques.

In stochastic analysis, a portable system with DPM control is modeled as a set of basic components: A service requestor (SR) models incoming events; the computation unit in the system which responds to external events and executes their corresponding tasks is modeled as a service provider (SP); and a power manager (PM) models the unit where power on-off control is provided. Incoming events from SR join the event queue (EQ) of events waiting to be serviced. The SP services events by executing corresponding tasks, and tasks waiting to be executed form the task queue (TQ). On the completion of a task, it is removed from the TQ and its corresponding event is removed from the EQ. The status of these two queues partly reflects the current state of the system.

Some of the previous stochastic researches about DPM systems try to find more sophisticated policies for the on-off power control [5]. Others give thorough analyses of different operation modes and/or transitions among these modes, so as to give accurate estimation about the average power consumption in the SP [6, 7]. All this research, however, takes the PM as cost-free in both energy and latency. Although the power dissipation in the PM is small compared with that in the SP, the former's total energy consumption may not be negligible given enough time accumulation since the PM never sleeps. Furthermore, the PM unit in many portable devices has extended to include some frequent routine executions such as task scheduling so as to give the energy hungry SP more time to sleep. A coprocessor will serve as a control unit (CU) and provide both Power Management (PM) and Task Management (TM) to the SP. In these cases, the power consumption in the coprocessor can not be simply ignored. Finally, different DPM policies influence the power consumption in the SP as well as in the PM. And the optimized policy is the one which can most reduce the average power consumption of the *entire* DPM system, not only the SP. All these reasons give us enough motivation to make new fine-grain power analyses about the DPM system with full consideration of the cost in the CU.

The nondeterministic events incoming and scheduling provides enough Markovian characteristics to the CU, and enable us to integrate the execution of the CU into our stochastic model [7] which previously only represents the states of the SP. In this case, the structure of the DPM system introduced before will be refined in Figure 1.

With the involvement of the CU execution, the event queue (EQ) is refined to two queues in Figure 1. External Event Queue (EEQ) represents the incoming events which just arrive in the DPM system. Events in EEQ will be removed from the queue as soon as the events have been processed by the CU. Events that have been processed in the CU form the Internal Event Queue (IEQ) and the task queue (TQ) keeps the same meaning as before. An event in the IEQ will be removed from the queue when the corresponding task is serviced in the SP.

The rest of this paper is organized as follows: Section 2 briefly reviews our previous models of two example DPM policies with full consideration of the cost in on-off mode transitions, and gives the corresponding power-latency analysis when the PM is taken as cost free. Section 3 will extend the models given in Section 2 to involve the execution in the CU in the state-flow diagram so as to give fine grain analysis about the DPM system. Conclusions and future work plans are given in Section 4.

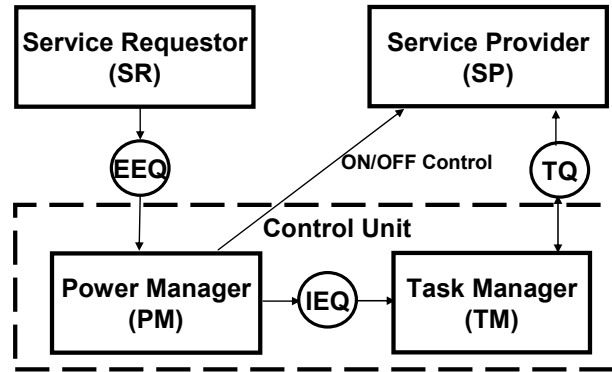


Figure 1: The structure of a DPM system

2. Previous Work

2.1 Greedy Policy

The simplest policy implemented in DPM systems is called *greedy* policy [6] or *eager* policy [8] which shuts a computation unit down as soon as the execution of the last task in the unit is completed and wakes up the unit as soon as a new event arrives. The on/off modes in the SP are called active/inactive modes respectively and the mode switching transitions between the two modes are called shutdown/wakeup respectively. Because the mode switching transitions bring both energy and latency cost to the system, our previous Markov model use additional states to describe the behavior of the SP in these transitions (Figure 2) so as to achieve more accurate performance analysis than other models [7].

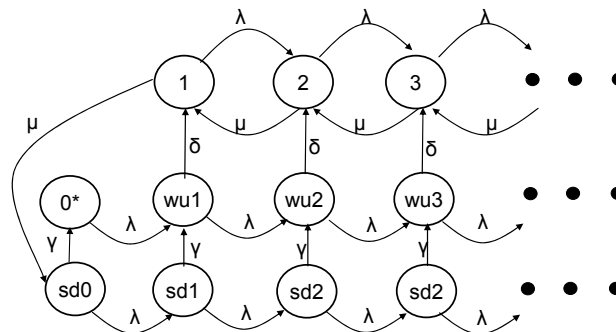


Figure 2: Markov model for greedy policy

Following tradition [3, 6, 8], λ and μ are the arrival rate of external events from the SR and execution rate of tasks in the SP respectively. When the SP is in inactive mode, the system is in state 0^* . With the arrival of a new event, the SP starts the wakeup transition which is represented by wakeup states $wu1$, $wu2$ and so on. In this

model, the wakeup and shutdown transitions are taken as Markov processes whose rates are represented by δ and γ respectively in Fig. 2. During the wakeup transition, the system starts in state $wu1$, and may then move to state 1 when the transition is completed without any other events coming. Otherwise, the system may move from $wu1$ to $wu2$ and even further if one or more event comes during the transition. In Figure 2, the length of EQ/TQ is set to infinite so as to make the model more general in implementation. After the wakeup transition is completed (e.g. at the completion of state $wu(n)$), the SP is activated and starts its execution of all the n tasks in the TQ. The system is now in state n .

When the execution of the last task in the TQ is completed (system leaving state 1), the SP starts a shutdown transition, described by shutdown states $sd0$, $sd1$ etc. If one or more event comes during the shutdown transition, the system enters a wakeup state immediately on completion of the shutdown transition. Otherwise, the SP starts sleeping and moves back to state 0^* .

2.2. A&F Policy

Considering the energy overhead of mode switching in the SP, a natural improvement is to reduce the mode switching frequency. The incoming events will first be accumulated in the EQ and the SP will be activated to batch processing when the length of EQ reaches a certain limit N (N is called accumulation limit). This policy, derived from the N -policy in queuing theory [9], is called *accumulation and fire* (A&F) policy in this paper because it is similar to the *integrate and fire* mechanism found in biological neural systems [10]. The model of A&F policy can be seen in Figure 3, and greedy policy can be regarded as a basic A&F policy with $N=1$.

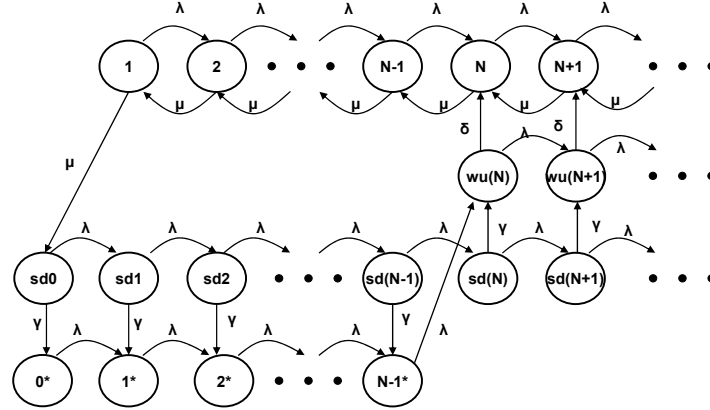


Figure 3: Markov model for A&F policy

According to the A&F policy, new events that arrive when the SP is sleeping cannot be executed until an accumulation limit N is reached. Therefore, states 0^* to $(N-1)^*$ are used in Figure. 3 to describe the change in the EQ when the SP is in its inactive mode.

If S_i , S_a , S_{wu} and S_{sd} are the sum of probability of all inactive, active, wakeup and shutdown states respectively, the average power consumption in the SP (\bar{P}) can be obtained with E2-1.

$$\bar{P} = S_i \times P_s + S_a \times P_w + S_{wu} \times P_{wu} + S_{sd} \times P_{sd} \quad (E2-1)$$

In E2-1, the power consumption when the processor is active, inactive, waking up and shutting down are P_w , P_s , P_{wu} and P_{sd} respectively. Different policies such as greedy and A&F policies will give different distributions among the four state groups so as to give different \bar{P} in the SP.

In our previous work, the stochastic models for both policies have been implemented using data from a realistic IBM portable Hard Disk Drive whose parameters are given in Table 1.

Table 1: IBM HDD parameters

Mode	Power (W)	Start-up Energy (J)	Transition time to Active
inactive	0	4.75	5s
active	1.9	0	0

In the analysis, the wakeup and shutdown transitions are assumed as symmetric, i.e. $P_{wu}=P_{sd}$ and the reciprocal of “Transition time to Active” in Table 1 are used as the value of the corresponding rates δ and γ . When the PM is taken as cost free, Figure 4(a) describes clearly that \bar{P} drops monotonically with the increasing of accumulation limit N .

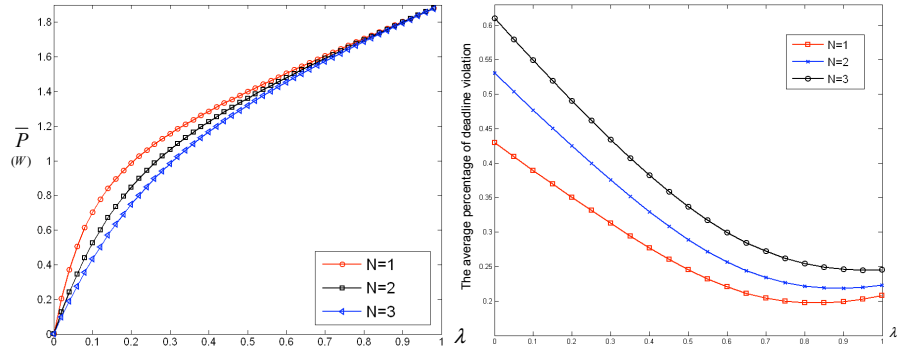


Figure 4: Power and Latency analysis of A&F policy

2.3. Latency Analysis

Although A&F policy can reduce the power dissipation especially when the accumulation limit is high, this is at the cost of longer latency. With soft deadlines widely used in chip design as the guideline for tasks' execution, the average probability of deadline violation for new incoming events is used in our previous work as the measurement of the latency of the DPM systems.

In order to differentiate power from probability, Q is used to represent probability and probability distribution while P is used for power consumption. The average probability of deadline violation (APDV) is calculated in the following way.

Suppose the SP is in (active) state n when a new event comes, the corresponding task is added to TQ and there are $n+1$ tasks so far waiting to be executed. The task corresponding to the new event will be executed only after the completion of the previous n tasks, assuming FCFS without losing generality (Non-FCFS execution sequences have no influence on the average probability of deadline violation). The latency in this case is the execution time of $(n+1)$ tasks in the SP. With the execution of each task following Poisson distribution, the execution of $(n+1)$ tasks follows Erlang distribution with parameters $n+1$ and μ . If t is the total time cost that the new task spent in the SP, $E_r(t; n+1, \mu)$ is the Erlang probability distribution function (pdf) and Q_n is the probability for the SP in state n , the average probability of deadline violation in this case can be expressed as E2-2:

$$APDV(n, DL) = Q_n \times \int_{DL}^{\infty} E_r(t; n+1, \mu) ds \quad (E2-2)$$

where DL is the mean deadline requirement of task execution.

Similarly, if the SP is in wakeup state $wu(n)$ when the new event arrives, the execution of the corresponding task must wait first for the completion of the wakeup transition and then the execution of the n tasks accumulated previously. If m is the fire moment of the SP, k is the time spent in the wakeup transition, and $P_{oisson}(k; \delta)$ represents the corresponding pdf of wakeup transition, the probability of deadline violation in this case can be expressed as E2-3:

$$APDV(wu(n), DL) = Q_{wu(n)} \times (1 - \int_0^{DL} E_r(t - m; n + 1, \mu) \int_0^m P_{oisson}(k; \delta) dk ds) \quad (E2-3)$$

The equations for inactive states and shutdown states are similar to E2-2 or E2-3. And the average probability of deadline violation of the SP is given in E2-4:

$$APDV(DL) = \sum_{n=0}^{N-1} APDV(n^*, DL) + \sum_{n=0}^{\infty} APDV(sd(n), DL) + \sum_{n=1}^{\infty} APDV(n, DL) + \sum_{n=N}^{\infty} APDV(wu(n), DL) \quad (E2-4)$$

Figure 4(b) gives the average probability of deadline violation for the example SP in Table 2. Here, the deadline is set to $10/\mu$ and it is clear that greedy policy has better latency performance than the A&F policy.

3. Fine grain Markov models for DPM systems

In order to better introduce the fine grain Markov models, we first give the architecture of the Control Unit in a DPM system when A&F policy is implemented (Figure 5).

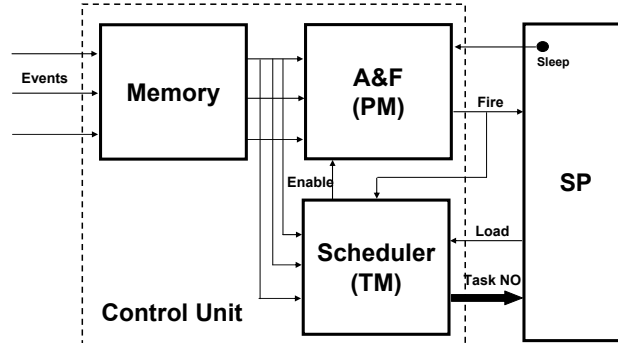


Figure 5: The Architecture of Control Unit

The *memory* part is used to save all events for later execution. In order to respond to stochastically incoming events, this part of CU has to be active all the time. The change in the memory will trigger the execution in the *Power Manager* to make new decision whether to activate the SP if the latter is inactive. When some prediction policy is implemented, this part has to be active all the time because the PM needs to update its history data for making more accurate prediction later. However, when the A&F policy is used, the PM can be totally shut down after the SP is activated because the A&F policy does not involve updating tasks. Therefore, different policies influence the power consumption not only in the SP, but also in the CU.

Furthermore, the latency in the CU will also be influenced by different policies. For example, if the greedy policy is used for the on/off control, the PM needs no more than a group of OR gates to make the activation decision to the SP and the latency caused by the PM in this case will be extremely small. However, when the A&F policy is used ($N > 1$), the PM circuits will become much more sophisticated. Adders will be needed to calculate the length of IEQ, and Arbiters will also be used to deal

with the metastability [11] caused by simultaneous arrival of events. The complexity in PM circuits increases both power dissipation and system latency.

The events in the memory will also be used by the *scheduler (TM)* part to find the corresponding tasks to be executed in the SP. Task scheduling will also be carried out in this part, and the number of chosen task will be sent to the SP when the task load command is received. No matter what kind of scheduling policy may be carried out in the TM, the time cost in every scheduling period varies from each other, which gives the TM strong Markovian feature in its execution.

Based on the architecture of the CU in Figure 6, we can extend the Markov models introduced in Section 2 to integrate the execution of the CU and form the new fine grain Markov models for DPM systems.

In section 2, the index of states in the Markov models represents the length of EQ (and TQ when the SP is active). With the decomposition of EQ to EEQ and IEQ, dual indexes will be used in the new fine grain models. The first index represents the length of EEQ and the second index reflects the length of IEQ (as well as TQ when the SP is active). For example, state (ij^*) describes the moment when the EEQ has i events waiting for execution in the CU and the IEQ has j events accumulated to activate the SP.

With the explicit representation of CU execution, the number of states in the Markov models increases from n to n^2 . This makes it difficult to show the full model in one figure. Therefore, each cell of Figures 7 describes one single tile of the fine-grain Markov model of A&F policy.

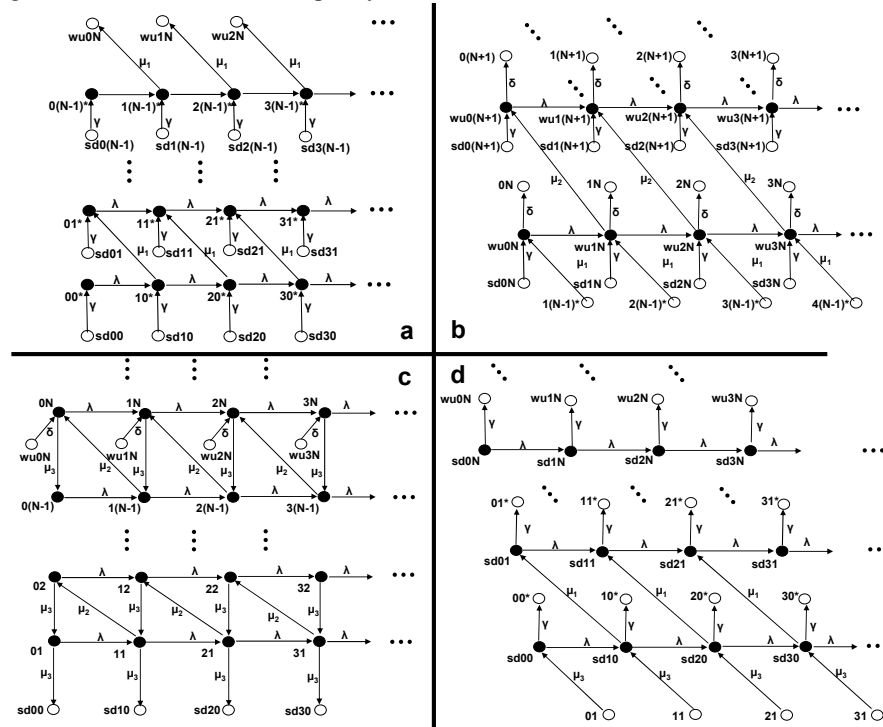


Figure 6: Tiles of fine grain Markov models (a) for inactive states (b) for wakeup states (c) for active states (d) for shutdown states

Figure 6(a) is about the inactive states of the new fine-grain model. The black nodes in the figure are the inactive states while the white nodes are the connected

states in other groups. Suppose one event comes when the system is in state 00^* , the EEQ becomes 1 and IEQ keeps 0 so the system moves to state 10^* . The arrival of the new event triggers the CU to start execution. If μ_1 is the execution rate of the CU (memory + PM in this case), the system may move to state 01^* when the processing in the CU can be finished before the arrival of the next event, or move to state 20^* when the new event comes before the completion of the PM execution. The more events that have been executed in the CU, the more events accumulate in IEQ. Given enough time, the system may reach state $i(N-1)^*$ ($i>0$), and the PM will activate the SP as soon as one more event is added to the IEQ.

The tile of wakeup states of the model is given in Figure 6(b). Once there are N events in the IEQ, the SP starts its wakeup process and the PM part of the CU will be shutdown to save power. At the same time, the scheduler (TM) starts to sort tasks for the execution in the SP according to the events in the IEQ. And μ_2 here is used as the execution rate of the CU (memory + TM in this case). If the system is in state wu_{ij} ($i>0, j>N$), its state movement has three possible directions: It may move to state $wu_{(i+1)j}$ if one new event comes, or move to state $wu_{(i-1)(j+1)}$ if the scheduling of $j+1$ tasks is complete, or move to active state ij if the wakeup process finishes.

Figure 6(c) describes the behaviour of the system when the SP is active. And μ_3 is used as the execution rate of the SP. The system in state ij ($i>0, j>0$) can move to one of three neighbour states: It may move to state $i(j-1)$ when the execution in SP is complete, or move to state $(i-1)(j+1)$ when the CU completes a new task scheduling, or move to state $(i+1)j$ when a new event arrives in the system. If the system moves to state $i1$, the SP will be shut down and move to shutdown state sd_{i0} when the SP execution is complete.

Figure 6(d) is the last tile of the system, which reflects the movement of the system when the SP is shutting down. The scheduler will stop working because no more service is provided in the SP and the PM is activated again to carry out A&F calculation. Therefore, the execution rate in the CU becomes μ_1 again. If the system is in state sd_{ij} ($j<N$), it will move to inactive state ij^* when the shutdown processing is complete. On the other hand, if the system is in state sd_{ij} ($j\geq N$), the SP will be activated immediately after the shutdown processing finishes because there are already N or more events in the IEQ.

If S_i, S_a, S_{sd} and S_{wu} keep the same meaning as in Section 2, E2-1 can also be used to calculate the average power consumption \bar{P}_{SP} (\bar{P} in section 2 because it assumes the PM is cost free) for the SP. \bar{P}_{CU} , as the average power consumption in the CU when the A&F policy is implemented, can be calculated in E3-1.

$$\bar{P}_{CU} = 1 \times P_M + (S_a + S_{wu}) \times P_{TM} + (S_i + S_{sd}) \times P_{PM} \quad (E3-1)$$

In E3-1, P_M , P_{TM} and P_{PM} are the power dissipation in the memory, TM, PM parts of the CU respectively. Because the memory will be always on, it will consume energy all the time. The PM consumes energy only when the SP is inactive or shutting down. And P_{TM} is weighted by S_a and S_{wu} because the TM only works when the SP is activating and activated. The average power consumption of the entire DPM system (\bar{P}) is the sum of \bar{P}_{SP} and \bar{P}_{CU} .

To demonstrate the usage of the fine grain model, the example SP in Section 2 is used again in the analysis. Table 2 gives parameters of the CU for different policies.

Table 2: Parameters of the CU

Greedy Policy				
μ_1	μ_2	P_M	P_{TM}	P_{PM}
1000	100	38mW	96mW	7.6mW

A&F Policy ($N=4$)				
μ_1	μ_2	P_M	P_{TM}	P_{PM}
100	100	38mW	96mW	66mW

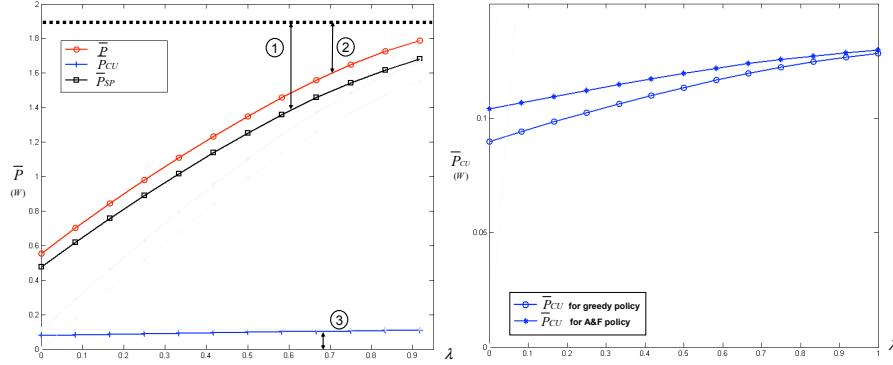


Figure 7: Fine Grain Power Analysis

In our analysis, we simply assume the maximum length of both IEQ and TQ is 7 and further coming events will be discarded if the IEQ is full. Figure 7(a) describes the power consumption in the example DPM system when greedy policy is implemented. When the straight line in the top of the figure stands for P_W , (1) is the gross power gain of the DPM system, and (2) is the power overhead paid in the CU and (3) is the net power gain of the system.

Figure 7(b) compares the power overheads in the CU when different policies have been implemented. It is clear that the A&F policy CU ($N=4$) consumes more power than the greedy policy CU. However, the overhead paid for the A&F policy can be worthwhile because the \bar{P} of the A&F policy is smaller than that of the greedy policy even when the power cost in CU is taken into consideration (Figure 8(a)).

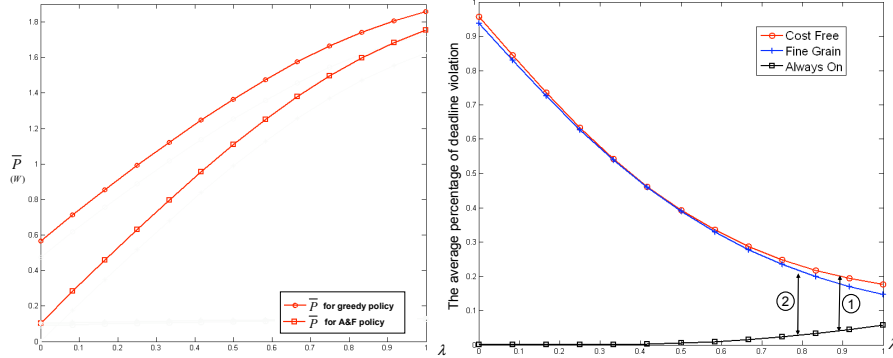


Figure 8: Fine Grain Analysis about A&F policy

The fine-grain model can also help analysis about the latency in the system. For example, as shown in Figure 8(b), when the curve in the bottom of the figure represents the average percentage of the deadline violation (DL is set to $10/\mu_3$) when the SP is always on, (1) and (2) serves as the gross and net extra latency cost by DPM control when the A&F policy ($N=4$) is implemented.

4. Conclusion and Future Work

Dynamic Power Management is a system level low-power technology which has been widely used in portable devices to reduce the average power consumption. This

paper presents a method to build fine-grain Markov models for DPM systems which integrate the execution of the DPM control into the state-diagrams. These models can be used to give more accurate power-latency analysis about the DPM system when the overhead of the control unit needs to be fully considered.

If the SP of the DPM system is some advanced processor, the execution of the PM is extended from simple on-off decisions to managing the switching among various modes provided by the SP. Markov modeling of such DPM systems when the PM is taken as cost free has been thoroughly explored in our previous work [12]. The models in this paper can be easily combined with the models in [12] to construct the fine-grain Markov Models for DPM system with multiple modes.

The fine-grain Markov model in this paper can also help hardware designers improve their circuits. The same high-level DPM policy can have various implementation circuits in the PM/TM with different power/latency parameters. And the analysis given in this paper can help hardware designers to compare the performance of these circuits as well as their influence on the entire system.

With DPM control, the portable devices are triggered by incoming events, such as data, signal and energy tokens. For the more explicit energy driven systems, the SP, which stands for the main processor in the system, can provide the execution only when enough energy is available. And the CU represents the energy-harvesting unit which is always alert to the environment and carries out the harvesting execution when the energy that can be collected is higher than that will be consumed in the task execution. Energy instead of data events will be accumulated in this unit and the activation of the main processor will depend on this accumulation. We believe that our current model, which is derived mainly with data and signal events in mind, may be further refined and modified to better suit energy driven systems. An immediate next task is to develop a more systematic and coherent representation of harvested energy as atomic events as well as example models of CUs from real-world energy harvesting systems.

5. Acknowledgement

This work is supported by the EPSRC through the STEP (EP/E044662/1) project.

6. References

- [1] L. Benini, G. de Micheli, "Dynamic Power Management: Design Techniques and CAD Tools", Kluwer Academic Publishers Norwell, USA, 1998.
- [2] L. Benini, A. Bogliolo, G. De Micheli, "A survey of Design Techniques for System-level Dynamic Power Management" IEEE Transactions on VLSI June 2000.
- [3] Q. Qiu, M. Pedram, "Dynamic Power Management Based on Continuous-Time Markov Decision Processes", DAC 1999.
- [4] T. Simunic, L. Benini, P. Glynn, G. de Micheli, "Dynamic Power Management for Portable Systems", the 6th annual international conference on Mobile computing and networking, Boston, 2000
- [5] Y. Tan, Q. Qiu, "A Framework of Stochastic Power Management using Hidden Markov Model", DATE 2008
- [6] Z. Ren, B.H. Krogh, R. Marculescu, "Hierarchical Adaptive Dynamic Power Management", IEEE Transactions on Computers, 2005.
- [7] Y. Chen, F. Xia, A. Yakovlev, "Stochastic Modeling Of Dynamic Power Management Policies And Analysis Of Their Power-Latency Tradeoffs", Technical Report, NCL-EECE-MSD-TR-2007-123, Newcastle University.
- [8] L. Benini, A. Bogliolo, G. A. Paleologo, "Policy Optimization for Dynamic Power Management", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1999.

- [9] K. Wang, H. Huang, "Optimal Control of a Removable Server in an M/Ek/1 Queuing System with Finite Capacity", *Microelectronics Reliability*, 1995.
- [10] G. Buzsaki, A. Draguhn, "Neuronal Oscillations in Cortical Networks", *Science*, June 2004.
- [11] A. Bystrov, D. Kinniment, A. Yakovlev, "Priority Arbiters," *ASYNC'00*, 128-137, Apr 2000, Eilat, Israel
- [12] Y. Chen, F. Xia, A. Yakovlev, "Stochastic Modeling Of Dynamic Power Management Policies And Analysis Of Their Power-Latency Tradeoffs" submitted to ICCAD08
- [13] A. Kansal, M. B. Srivastava, "An environmental energy harvesting framework for sensor networks", *ISLPED'03*, Seoul, Korea.

Distributed Duty-cycle Management for Dependable Wireless Sensor Networks

Jin Wu and Zhili Sun*

Abstract

It is believed that the deployments of Wireless Sensor Network (WSN) have great long-term economic potential, ability to transform our lives, and pose many new system-building challenges. A typical application domain of WSN is by placing a vast number of multifunctional sensor nodes over a field to sense and collect surrounding environment data. Beside energy conservation concerns, another major challenge for its real deployments is its reliability issue, more precisely the sensing resolution (this being the main service quality we addressed in this paper). In many cases, redundancy of sensors is being made to provide higher sensing resolution when single sensing device is with lower ability. Certain coverage of sensors is required in order to guarantee the sensing data with acceptable degree-of-truth. However, since sensor nodes are mostly powered by batteries and required to remain in inactive state for the longest possible time duration due to save energy. Balancing the trade-off between energy-efficiency and sensing quality is a rich area because sensor deaths and sensor replenishments make it difficult to specify the optimum number of sensors that should be activated and sending information at any given time. Through literature survey, we discover that current solutions toward this problem fell into some limitations in configurations or deployments. In this paper, we present a concept for improving the overall performance of the WSNs through local collaborations of neighbour nodes, and provide a more efficient duty-cycle management solution. A framework for distributed duty-cycle management is given, and a control algorithm is generated from the framework. Simulation shows that the new method does work in WSN environments, and gives good results.

Keywords: Wireless Sensor Networks, Duty-cycle, Distributed, Resolution

1 INTRODUCTION

Recent advances in Micro-Electro-Mechanism Systems (MEMS), Wireless Communications, and Embedded Systems made it possible in manufacture cheap, small, and energy-efficient multifunctional sensors with the ability of sensing environments, processing data, and transmitting information. A sensor network can be formed by deploying hundreds and thousands of multifunctional sensors within the region of interest

* Centre for Communication System Research (CCSR), University of Surrey, UK, {jin.wu; z.sun}@surrey.ac.uk

to perform certain sensing and networking tasks. Sensor Networks have generated flurry of research activity because of variety of applications [1, 2]. Some of the applications envisioned for such networks include but are not limited to targets detection, object tracking, environment surveillance, intrusion detection, and etc. [3, 4]. Many of these applications involve large-scale sensor networks, where a large number of sensors are deployed in a vast geographical area, and operating simultaneously in separate geographical spots.

WSNs are placed in the field to collect data of interest for applications. Then, the ability of data collection of the WSN directly affects the performance of applications. Service quality in WSN involves a wide verity of criteria. A special service quality parameter to mean the resolution of sensor networks is a unique parameter typically measures the performance for WSN in terms of the data gathering ability. Different from conventional communication networks that only provide the service of duplicating information in bits from one end of the network to the other end of the network, sensor networks involve the ability of data collection, whereby the collection ability also needs to be ensured. Reference [6, 7] directly defines QoS to mean sensor network resolution. Specifically, depending on the different stimuli present in the sensor network, it is defined as the optimum number of sensors sending information toward information-collecting sinks. This is a very important issue, because in any sensor network we want to accomplish two things: 1) maximize the lifetime of the sensor network by having sensors periodically power-down to conserve their battery energy, and 2) have enough sensors powered-up and sending packets toward the information sinks so that enough data is being collected when stimuli presents. Note that the information sinks need a certain amount of information gathered from the different sensors, but sensors in close proximity to each other allow many of those sensors to be powered-down.

This is the management problem we address. A most straight forward solution to this problem is the static setting whereby the duty cycle of each sensor node is settled in a fixed way before its deployment. It is widely used approach for nodes' periodically powered off in the WSN. However, the duty-cycle management is a rich research area because sensors are always placed in the sensing field in random fashions with redundancy. Sensor deaths (e.g., as a result of damage or battery failure) and sensor replenishments make it difficult to control the optimum number of sensors that should be activated and sensing the field at any given time [5, 6]. Therefore, adaptive mechanisms should be applied to allow the running state of the sensor nodes cope with the randomness brought by the uncertainties from the sensing field. This issue is firstly described in ref [6] and reputed as the QoS problem in WSN. Then, some improvements are given in ref [7, 12]. However, a significant weak point for those solutions is that two assumptions are made where 1) broadcast channel exists for collection point to all nodes, and 2) sensor nodes are able to acknowledge the information for collection point even when it is powered off for energy saving. The above two assumptions are not supported by mainstream sensor node equipments. In this paper we present a distributed control algorithm for duty-cycle management to improve the QoS of Wireless Sensor Networks. We consider the sensor network can operate under the following model. Sensor nodes are distributed across the sensing field and simultaneously operating. Each sensor nodes swap

within two states, active and sleep. Sensor node itself has to decide when in active (or sleep) state. A control algorithm runs on each sensor node to determine the operation state. The simulations show that the 'borrowed' well fits the reality of duty-cycle control in Wireless Sensor Networks. The reminder of this paper is organised as follows. Section 2 gives the related works and model definitions for this research. Section 3 gives the concept of doing duty-cycle management in a distributed manner. Section 4 gives a detailed description of the control algorithm for duty cycle. Simulation and conclusions are given in Section 5 and Section 6.

2 BACKGROUND

The study of wireless sensor networks is still a burgeoning field, many aspects of sensor networks, such as routing, preservation of battery power, adaptive self-configuration, etc., have already been studied in previous papers. Ref. [8] might be the earliest work to the present study as it actively probes the question of QoS that the base stations are receiving from the sensors. However, it defines QoS as total coverage in a static fashion. That is, it does not allow a data sink to dynamically alter the QoS it is receiving from the sensors, depending on varying circumstances.

Reference [6] proposed a solution that uses the idea of allowing the base station to communicate QoS information to each of the sensors using a broadcast channel and then use the mathematical paradigm of the Gur Game to dynamically adjust to the optimum number of sensors. The result is a robust sensor network that allows the base station to dynamically adjust the number of sensors being activated, thereby controlling the resolution of QoS it receives from the sensors, depending on varying circumstances. This research attracts some research attentions and some new papers [7, and 12] can be found in the literature to extend the idea. However, the limitation for this Gur Game based algorithm and its variations is that two unreasonable assumptions are given, where 1) a broadcast channel is needed from base station to all sensor nodes, and 2) Sensor nodes need to reply to the request from the base station even when it is in sleep state. These algorithms can hardly be deployed to real sensor works without a radically change.

In this paper, Placement Model, Sensing Model, and Converge Measures are defined as follows, respectively. In the rest of this paper, a commonly used sensor placement model is applied. This model has been used by many researchers, e.g. in ref. [2]. Large number sensors are randomly placed over a two-dimensional geographical region. It is also assumed that the locations of sensors are uniformly and independently distributed in the region. Such a random initial deployment is desirable in scenarios where priori knowledge of the field is not available. Also, the random deployment can be the direct result of certain deployment strategies. Based on this assumption, the locations of sensors can be modelled by a stationary two-dimensional Poisson point process. Denote the density of the underlying Poisson point process as λ , which is measured by the number of sensors per unit area. The number of sensors located in a region A , $N(A)$, follows a Poisson distribution of parameter $\lambda\|A\|$, where $\|A\|$ represents the area of the region [2].

$$P(N(A) = k) = \frac{e^{-\lambda\|A\|} (\lambda\|A\|)^k}{k!} \quad \dots (1)$$

In this paper, for the sake of simplicity, the Boolean sensing model is being used. The Boolean sensing model has been widely used in many researches. In the Boolean model, each sensor has a certain uniform sensing range, r . A sensor can only sense the environment and detect phenomenon within its sensing range. A location is said to be “covered” by a sensor if it lies within the sensor’s sensing range. The degree of coverage is defined by the coverage density. It is defined as, $f_c(p)$, the positive integer for sensors’ number by which a particular point p within the sensing field is covered with. The Coverage Density represents redundancy level of sensor deployment for a certain point in the detection area. Note that the definition of a location being covered depends on the specific sensing model under consideration. The Boolean sensing model is considered in this paper, where a location is covered if it is within the sensing area of a sensor. If there are n sensors got the ability to detect the event that takes place at the point p , then the value of $f_c(p)$, in terms of resolution, equals to n . Therefore, if there is an event takes place at the point p , basically, n sensors will be able to claim detection for this event. However, some sensors in the sensor network might face some problems in terms of miss detection and false detection, so the detection reported by sensors might not exactly equals to n . Detail performance of n will be discussed in Section 5.

3 DISTRIBUTED DUTY-CYCLE MANAGEMENT CONCEPT

As discussed in Section 1, most sensor network applications require service quality guarantees in terms of data resolutions, suggesting the need for redundancy in the sensing ability. However, the energy supplies for sensor nodes, in many cases, are not without limitations. So for the data sensing ability concerns, sensor nodes should remain in active state to collect environment data from the field, whereas sensor nodes are expected to periodically powered-off for the energy conservation concerns. A clear trade-off exists for every sensor nodes in the field, on one hand it should be activated to provide more sensing abilities; on the other hand, it should be kept in inactive state for longest possible time to save energy. So a mechanism needs to be proposed to balance the aforementioned requirements. Solutions toward such problem can be done in two different ways, proactive and reactive. A proactive solution means the sensor nodes will be configured with the duty-cycle parameter before its deployment. However in order to provide the parameter value, a complete set of knowledge, such as position of sensor nodes, sensing range, energy consumptions, etc. should be given. Such knowledge is hard to acquire in many cases, especially when the sensor nodes are randomly deployed in locations and sensor nodes will die out and replenish, there is no way acquire such information before its real deployment. So, the reactive solution is proposed by providing a mechanism that allows the sensor network to adapt to the environment. In this solution, the duty-cycle parameter is set after its deployment, and might be reconfigured during the running process. There exist two approaches for the reactive solution, centralised and distributed. For centralised configuration, a computation device acting as server will collect knowledge related to the parameter setting, and then configure the sensor nodes in real-time. Limitations for this approach are that sensor nodes are strictly resource intensive devices. Running a central configuration approach will bring additional resources in information transmissions and

computation.

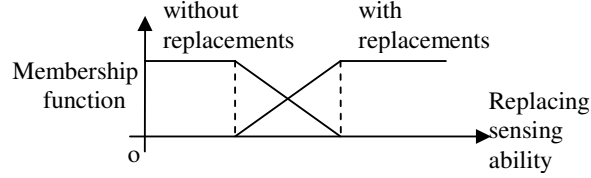


Figure 1. Concepts of Distributed Duty-cycle Management

In this paper, we adopt a distributed reactive duty-cycle management mechanism. In this mechanism, sensor node probes surrounding environments and configures its duty-cycle through greedy local decisions. A guideline is applied where the duty-cycle should be decreased when the probe detects enough similar sensor nodes is in active state, or vice versa. The concept can be given in Figure 1. Replacing sensing ability can be given by exchanging information with neighbour nodes. Membership function is given by applications. A simple example is given in the following section to explain the concept.

4 DUTY-CYCLE MANAGEMENT ALGORITHM IN SENSOR NODES

Suppose we have a collection of n sensor nodes, M_1 through M_n , with unified sensing ability placed over a sensing field with area of S . Sensor nodes are placed and operating following the definition in Section 2. This section gives an example of the concept mentioned in Section 3. A detailed duty cycle management algorithm is given. For the sake of simplicity, it is assumed that all sensor nodes have similar hardware and software specifications and configurations, and they are placed over a flat screen in quite radio environment. As defined in Section 3, every node is running independently. In order to have every node periodically power-down to conserve its battery energy, sensor nodes need to compute what time to sleep and what time to wake up. For any node i , it is awake for the first time deployment, suppose the time for the h^{th} sleep is $t_{sleep}^i|_h$, and the time for the h^{th} wake up is $t_{wake}^i|_h$. Then, the h^{th} sleep interval can be represented as

$$\Delta t_{ivl}^i|_h = t_{wake}^i|_h - t_{sleep}^i|_h$$

When in active state, the sensor node will emit a beacon signal through the radio channel for every Δt_{beacon} as the heart beating. Without loss of generality, we can define

$$\Delta t_{beacon} \gg Z_{beacon} / Rate,$$

where Z_{beacon} and $Rate$ represent for the packet size of beacon and the transmission rate of radio channel. The beacon density, $n^i(t)$, measures the probability of receiving a beacon for node i at time t . Then, it can be easily discovered that the resolution of a event takes place at point x , $x \in S$, is

$$f_c(x, t) = n^x(t) \cdot \Delta t_{beacon} \cdot \left(\frac{R}{r}\right)^2 + 1 \quad \dots(2)$$

where R and r are the range for sensing and transmission respectively. Defined the minimum threshold of $f_c(x, t)$ as f' , then the minimum threshold of beacon density $n^x(t)$ can be represented as

$$n^x(x) = \left(\frac{r}{R}\right)^2 \cdot \frac{f'_c - 1}{\Delta t_{beacon}} \quad \dots(3)$$

Based on the above analysis, it can be considered as when the incoming beacon probability is lower than $n^x(x)$, it means that the number of nodes that cover the point x are too small to provide certain degree of resolution, or *vice versa*. Therefore, the algorithm that controls the sleep interval can be given as follows.

```

 $T_{wake}^{i|_0} = 0;$ 
 $T_{sleep}^{i|_0} = 0;$ 
 $n = 1;$ 
if  $((t_{now} - last\_update) > update\_itvl)$ 
{
  if  $((((t_{now} - t_{wake}^{i|_{n-1}}) / \Delta t_{beacon}) * x) > Max_{th})$ 
  {
     $t_{sleep}^{i|_n} = t_{now};$ 
     $t_{wake}^{i|_n} = t_{now} + Max\_sleep;$ 
  };
  if  $(((((t_{now} - t_{wake}^{i|_{n-1}}) / \Delta t_{beacon}) * x) < Max_{th}) \& (((t_{now} - t_{wake}^{i|_{n-1}}) / \Delta t_{beacon}) * x) > Min_{th}))$ 
  {
     $t_{sleep}^{i|_n} = t_{now};$ 
     $t_{wake}^{i|_n} = t_{now} + Max\_sleep * (((t_{now} - t_{wake}^{i|_{n-1}}) / \Delta t_{beacon}) * x - Min_{th}) / (Max_{th} - Min_{th});$ 
  };
  if  $(((((t_{now} - t_{wake}^{i|_{n-1}}) / \Delta t_{beacon}) * x) \leq Min_{th}))$ 
  {
     $t_{sleep}^{i|_n} = +\infty;$ 
     $t_{wake}^{i|_n} = t_{now};$ 
  };
  if  $t_{sleep}^{i|_n} \leq t_{now}$ 
  Set_wake( $t_{wake}^{i|_n}$ );
   $x = 0;$ 
   $n++;$ 
   $last\_update = t_{now};$ 
  Sleep;
};
 $Min_{th} = (r/R)^2 * (f'_c - 1);$ 
 $Max_{th} = Min_{th} + a * (N - Min_{th});$ 

```

Figure 1. Duty-cycle Control Algorithm for QoS Control

In the algorithm shown in Figure 1, t_{now} is current time, and the $update_itvl$ is the time slot for update interval. $\alpha \in (0, 1)$ is a parameter related to the stability of system.

5 PERFORMANCE ANALYSIS

The detailed analysis for the setting of this parameter is overloaded with details. However, the brief description of the running state of the algorithm can be given as follows. The relationship between $f_c(x,t)$ and Δt_{ivl} can be represented as follows. Suppose at time t , as a total of k nodes are able to sense the point x , u of which are in active state and others are in sleep state. If a steady state is reached, then

$$u \cdot \frac{\Delta t_{ivl}}{\Delta t_{ivl} + \Delta t_{active}} = (k - u) \cdot \frac{\Delta t_{active}}{\Delta t_{ivl} + \Delta t_{active}} \quad \dots(4)$$

where Δt_{active} is the active period when a sensor node once wakes up. From the equation (4), the beacon density can be represented as

$$n = \frac{k}{\Delta t_{active} + \Delta t_{ivl}} \quad \dots(5)$$

The equation (5) plots as the dot line curve in Figure 2. The control algorithm is plotted as the real line curve

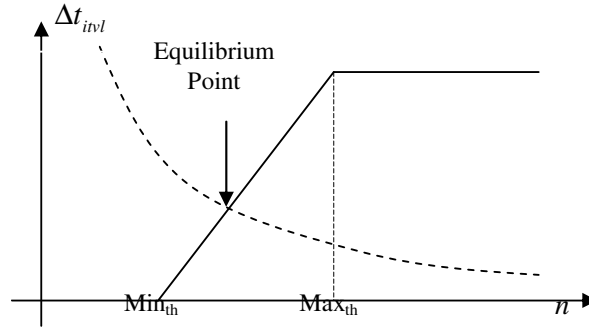


Figure 2. The Operation State of Duty Cycle Management Algorithm

When the beacon density of a sensor node is lower than the Min_{th} , it means that the active node nearby is not enough, therefore the sleep interval should approach to 0 in order to bring up the resolution. When the beacon density is between the Min_{th} and Max_{th} , it means enough sensing ability has been activated nearby, so the sleep interval can be linearly increase as the beacon density increases. When the beacon density is higher than the Max_{th} , it means that active sensor nodes nearby are far more than those required. Then, the sensor node should set with the longest sleep interval. The selection of Max_{th} depends on the loss parameter defined in Figure 2. The detailed set of Max_{th} is too much in details and exceeds the scope of this paper. A simple principle of setting Max_{th} is that higher value of Max_{th} will increase the stability of the control system but harm the convergence speed, and lower value of Max_{th} will do *vice versa*. In this paper, we set $\text{Max}_{th} = 2\text{Min}_{th}$.

6 SIMULATIONS

Since the experiment requires a large number of sensor nodes to test the affect of duty-cycle management, due to limitations in hardware, the Simulation is done on Matlab

in order to verify the control algorithm we proposed. A simple script is programmed to simulate the function of the algorithm. We compare the newly proposed algorithm with the well-known algorithm 'GUR' [6] in terms of steady-state performance, and standard deviation of coverage density. It is proved through simulations that the newly proposed algorithm does provide substantial improvements in performances. The detailed settings of the simulation experiments are given as follows. There have 250 sensors in the 100×100 sensing field with no sensor failures or renewals. All of them are networked with wireless channel without the consideration of routing problems. Each sensor is on active state as its initial state. R and r are set as 10 and 20 respectively. The desired resolution is set as 2. The simulation result for steady-state performance is as shown in Figure 3.

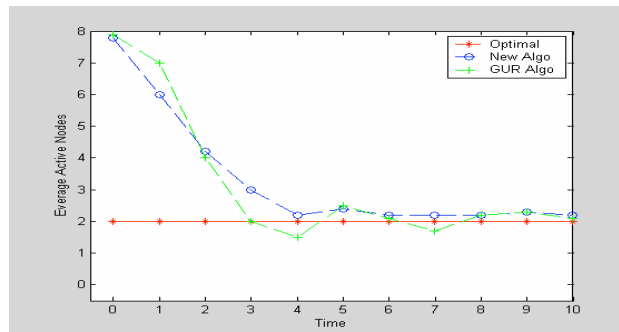


Figure 3. Steady-state Performance.

As one can see from Figure 3, both the Gur algorithm and the newly proposed algorithms are driving the resolution approaching the optimal value after a period of time.

Then, we randomly select 50 nodes in the sensing field and measure their coverage density for every second. The deviation of those 50 nodes is given in as follows.

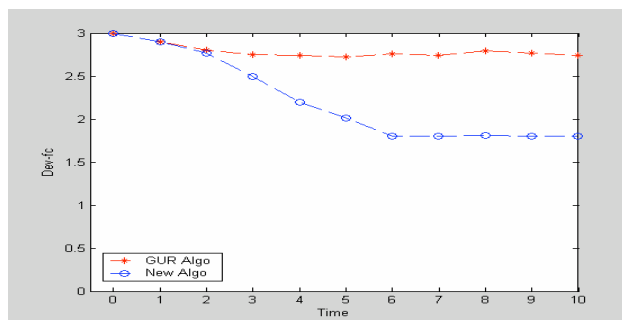


Figure 4. Deviation Performance

As observed in Figure 4, the deviation for the newly proposed algorithm is clearly lower than the Gur algorithm, whereby clear evidence is given that the distributed algorithm does give a higher quality of performance in terms of fairness. From simulations, it shows

that the new control algorithm does functioning, and it is able to provide some sort of QoS.

7 CONCLUSIONS

Sensor networks are an exciting area with very real applications in the near future. Although many aspects of sensor networks have been studied before, quality of service (QoS) for sensor networks remains largely open. In this paper, we present an idea of using the duty-cycle control algorithm running on each sensor node to balance the trade off between energy consumption and resolution. It has been proved by simulation experiment that even without using such critical assumptions as ref [6] did, we still can control the Wireless Sensor Networks will to achieve some sort of QoS. It is believed that our newly proposed control method is effective, and has significant advantage against the method in literature. In this paper we have reported the results of a study over a very small portion of the design space, and in this area, much future work remains.

References

1. S. Adlakha, M. Srivastava. Critical Density Thresholds for Coverage in Wireless Sensor Networks, IEEE Wireless Communications and Networking, 2003. WCNC 2003, pp1615-20 vol.3, March 2003.
2. B. Liu, D. Towsley. A Study of the Coverage of Large-scale Sensor Networks. 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp475-83, 2004.
3. A. Mainwaring, et. al. Wireless Sensor Networks for Habitat Monitoring, First ACM International Workshop on Wireless Sensor Networks and Applications, 2002.
4. A. Baptista, et. al. Environmental Observation and Forecasting Systems: Vision, Challenges and Successes of a prototype. ISEIS'2003, 2003.
5. W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," Proc. 5th ACM/IEEE Mobicom Conference, Seattle, WA, August 1999.
6. R. Iyer and L. Kleinrock, QoS Control for Sensor Networks, ICC'03.
7. Jeff Frolik. QoS Control for Random Access Wireless Sensor Networks, Proc WCNC 2004, pp1522-7, 2004.
8. S. Meguerdichian, K. Farinaz, P. Miodrag, M. Srivstava, "Coverage Problems in Wireless Ad Hoc Sensor Networks," ICC-IEEE International Conference on Communications, 2001.
9. D. Tian N. Georganas. A Coverage-preserving Node Scheduling Scheme for Large Wireless Sensor Networks. 1st ACM International Workshop on Wireless Sensor Networks and Applications, pp32-41, 2002.
10. F. Ye, et al. Peas: A Robust Energy Conserving Protocol for Long-lived Sensor Networks,
11. S. Shakkottai, et. al. Unreliable Sensor Grids: Coverage, Connectivity, and Diameter, IEEE Infocom, 2003.
12. J. Kay and J. Frolik. Quality of Service Analysis and Control for Wireless Sensor Networks, In Proc. of 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp359-69, 2004.

Validation of Large Zoned RAID Systems

Abigail S. Lebrecht Nicholas J. Dingle William J. Knottenbelt *

Abstract

Building on our prior work in [8], we present an improved model for for large partial stripe following full stripe writes in RAID 5. This was necessary because we observed that our previous model tended to underestimate measured results. To date, we have only validated these models against RAID systems with at most four disks. Here we validate our improved model, and also our existing models for other read and write configurations, against measurements taken from an eight disk RAID array.

1 Introduction

Choice of RAID level can critically affect the performance delivered by a storage system. Therefore the ability to predict RAID performance for RAID systems of different sizes and configurations is crucial. Modern Service Level Agreements require effective performance predictions must provide the ability to reason not only about mean response times, but also higher moments and percentiles of response time.

Previous RAID models [4, 7, 11, 16, 17] approximate only the mean response time of the system. All RAID models that we develop approximate the full response time distribution, from which moments and percentiles can be calculated. In [9], we introduced analytical queueing network models of RAID 01 and 5, the two most commonly used RAID levels. We extended these models in [8] through the use of a multiclass queueing network to allow heterogeneous workload streams of read and write requests. In both cases, we validated these models against device measurements from a real-life RAID system. This demonstrated the accuracy of our models and also suggested some areas in which they could be more representative.

In this paper, we present improvements to our existing RAID 5 models for those cases where large partial stripe writes follow one or more full stripe writes. We demonstrate the accuracy of this model and our other models [9, 8] by validating them against a real eight disk RAID system. This is an improvement over [9], where RAID 01 was only validated for one size of request (2 blocks) under the same load on a 4 disk array. Furthermore, constraints of validating on a four disk array meant that there was only one validation for each size of RAID 5 write request (small partial, large partial and full-stripe). On an eight disk array there are three different configurations representing each of small and large partial stripe writes.

In [8] the model was extended to allow mixed streams of read and write requests. Specifically we studied the cases of a request stream made up of the same amount of read and write requests, and weighted in either direction with proportions of 75% and

*Department of Computing, Imperial College London, 180 Queen's Gate, London SW7 2BZ, United Kingdom. Email: {asl102,njd200,wjk}@doc.ic.ac.uk

25% for each. We also studied the four disk array under different loads and for a wide range of request sizes for all types of arrival streams and observed certain trends that we discussed in that paper. By carrying out similar validations on an eight disk array, we hope to see to what degree the previous observations were anomalous, and to discuss possible reasons and model changes if the trends are still visible.

The remainder of this paper is organised as follows: in Section 2 we include a summary of our existing models, before presenting the improved RAID 5 large partial stripe write model in Section 3. We then validate our models more comprehensively by comparing the analytical results with measurements from an eight disk array for a variety of request sizes in Section 4.

2 RAID Model

Our RAID model is developed in a bottom-up hierarchical fashion. We begin by modelling each disk drive in the array as a single M/G/1 queue. An important subtlety that needs to be taken into account in the service time distribution is that modern disks are zoned, with more sectors on the outer tracks than inner tracks. Therefore, a random request is more likely to be directed to a sector on an outer track, and it is also faster to transfer data on a track closer to the circumference than the centre of the disk.

The service time density of an access to a random location on a single zoned disk is the convolution of the seek time, rotational latency and data transfer time probability density functions. We use the seek time and rotational latency distributions defined in [19] and the data transfer time distribution from [9]. We denote the random variables of seek time, rotational latency and k -block transfer time as S , R and T_k respectively. The response time distribution of the M/G/1 queue is obtained by numerically inverting [1] the corresponding Pollaczek-Khintchine transform equation [6].

We then abstract the RAID as a fork-join queueing network [3] of M/G/1 queues. In an N -queue fork-join network, (see Fig. 1), each incoming job is split into N subtasks at the fork point. Each of these subtasks queues for service at a parallel service node before joining a queue for the join point. When all N subtasks in the job are at the head of their respective join queues, they rejoin (synchronise) at the join point.

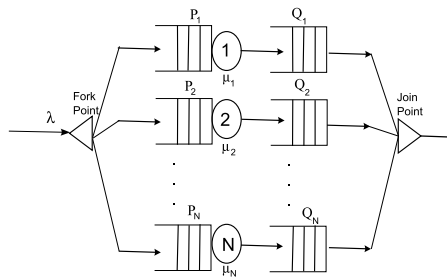


Figure 1: Fork-join queueing model

It is difficult, however, to model job response times in a fork-join synchronisation analytically. Indeed, exact analytical results only exist for the mean response time of a two server system consisting of homogeneous M/M/1 queues [12]. Approximations for mean response times for M/M/1 and M/G/1 fork-join queues are more abundant [12, 14, 15, 16, 18] but such results do not permit higher moments or full response time

distributions to be calculated. Therefore, we have previously presented [9] an approach using the maximum order statistic [5, 10] to derive an approximation to the cumulative distribution function of a fork-join queue's response time. This was inspired by [7], which defined an approximation of the fork-join queue that enables the calculation of both the mean and further moments of response time.

The standard fork-join network directly models the behaviour of a RAID system in only a small number of cases (e.g. full stripe I/O operations in RAID 0). Consequently, the fork-join model must be tailored to support the full range of I/O access patterns that occur when performing read or write operations of different sizes on different RAID levels. Our initial model is designed to accept a homogeneous stream of I/O requests of a given size and type (RAID 01 or 5, read or write). We further assume that all the service time distributions on all disks are identically distributed. For the sake of notational simplicity, let $W_d(t, \gamma, \frac{1}{\mu})$ define the cdf of the response time distribution of a single M/G/1 queue (disk), γ is the arrival rate at an individual disk and μ is the mean service rate. We assume there are n disks in the array and that the arrival rate of logical I/O requests to the disk array as a whole is λ . In [9], these models are introduced and we summarise them here.

The cdf of the response time for a b -block read from RAID 01 is:

$$W_{\text{read}}(t) = \begin{cases} \left(W_d\left(t, \frac{\lambda b}{n}, E[R] + E[S] + E[T_1]\right) \right)^b & \text{if } b < n \\ \left(W_d\left(t, \lambda, E[R] + E[S] + E\left[T_{\frac{b}{n}}\right]\right) \right)^n & \text{otherwise} \end{cases}$$

Similarly, the cdf of the response time of a b -block mirrored write to RAID 01 is:

$$W_{\text{write}}(t) = \begin{cases} \left(W_d\left(t, \frac{2\lambda b}{n}, E[R] + E[S] + E[T_1]\right) \right)^{2b} & \text{if } 2b < n \\ \left(W_d\left(t, \lambda, E[R] + E[S] + E\left[T_{\frac{2b}{n}}\right]\right) \right)^n & \text{otherwise} \end{cases}$$

The cdf of the response time for a b -block read from RAID 5 is:

$$W_{\text{read}}(t) = \begin{cases} \left(W_d\left(t, \frac{\lambda b}{n}, E[R] + E[S] + E[T_1]\right) \right)^b & \text{if } b < n \\ \left(W_d\left(t, \lambda, E[R] + E[S] + E\left[T_{\frac{b}{n}}\right]\right) \right)^n & \text{otherwise} \end{cases}$$

Due to parity calculation a RAID 5 write request is modelled differently for different sized requests. The simplest RAID 5 write request is one which consists only of a number of complete stripes (i.e. $b \bmod (n - 1) = 0$). In this case, computation of parity does not require pre-reading of existing data and so the only operation is to write to all disks. The cdf of request response time is therefore defined as:

$$W_{\text{write}}(t) = \left(W_d\left(t, \lambda, E[R] + E[S] + E\left[T_{\frac{b}{n-1}}\right]\right) \right)^n$$

A write request involving a partial stripe write will consist of two separate requests. The first is a pre-read for the calculation of the new parity. Then when all the parity pre-reads are completed and the new parity calculated, the second request, a partial stripe and new parity write request, is issued. Therefore, for partial stripe writes, we define a mean service time and density as the average of the service time (mean or density) of the parity pre-read and write request that follows. We note that these two subtasks are not independent. Indeed, we assume that they are highly dependent, and therefore the overall response time of a write request will be:

$$W_{\text{write}}(t) = P(2W \leq t) = P\left(W \leq \left(\frac{t}{2}\right)\right)$$

If a request consists of $b < \frac{n-1}{2}$ blocks (i.e. a small partial stripe write), the pre-read involves reading the old parity and data that will be replaced for parity calculation, then writing the new data to the same disks. The write will start after the last pre-read completes, so one disk will need to complete a full rotation (R_{max}) to return to the same sector. The overall response time cdf is therefore:

$$W_{write}(t) = \left(W_d \left(\frac{t}{2}, \frac{2\lambda(b+1)}{n}, \frac{(2b+1)(E[R] + E[S]) + R_{max}}{2(b+1)} + E[T_1] \right) \right)^{b+1}$$

The pdfs of both seek time and rotational latency must then be scaled accordingly to conform to the mean service times above:

$$f'(t) = \begin{cases} \frac{1}{2(b+1)} & \text{if } t = 0 \\ \frac{2b+1}{2(b+1)} f(t) & \text{otherwise} \end{cases}$$

where $f(t)$ is the probability density function of seek time or rotational latency.

For a large partial stripe write, $\frac{n-1}{2} \leq b < n-1$, the new parity is calculated by pre-reading from the disks that will not be written to and XORing it with the new data. Therefore at some point in the request each disk in the array is written to. The cdf of request response time is:

$$W_{write}(t) = \left(W_d \left(\frac{t}{2}, \lambda, E[R] + E[S] + E[T_1] \right) \right)^{n/2}$$

If $b > n-1$ and $0 < b \bmod (n-1) < \frac{n-1}{2}$, at least one full stripe write will occur followed by a small partial stripe write. The first request consists of the full stripe writes and the pre-read, and the second is the partial stripe write. Let $b_{mod} = b \bmod (n-1)$. The cdf of request response time is:

$$W_{write}(t) = \left(W_d \left(\frac{t}{2}, \frac{\lambda(n + b_{mod} + 1)}{n}, \frac{(n + b_{mod})(E[R] + E[S]) + R_{max}}{n + b_{mod} + 1} + E[T_{\frac{k}{2} + \frac{b_{mod}+1}{n}}] \right) \right)^{\frac{n + b_{mod} + 1}{2}}$$

In [9], our RAID models assumed homogeneous arrival streams. In [8] we used multiclass queues to generalise these models for heterogeneous streams composed of both reads and writes. The resulting request response time cdf of a RAID system with a mixed arrival stream of read and write requests is defined as:

$$W(x) = p_{read} W_{read}(x) + (1 - p_{read}) W_{write}(x)$$

where p_{read} is the probability that a request is a read.

We note that the arrival rate to the disk array defined for each type of request must be modified to take the combined stream into account. For RAID 01 the arrival rate at each disk becomes:

$$\frac{\lambda(p_{read} \min(b, n) + (1 - p_{read}) \min(2b, n))}{n}$$

On RAID 5, the arrival rate at each disk is:

$$p_{read} \lambda \frac{\min(b, n)}{n} + (1 - p_{read}) \gamma$$

where γ is the arrival rate at each disk in the array in the case that $p_{read} = 0$.

3 Improved Large Partial Stripe Write Model

Our validation work in [8] suggested that the model for large partial stripe following full stripe writes (where $\frac{n-1}{2} \leq b \bmod (n-1) < n-1$) could be improved as it tended to underestimate the measured results. Furthermore, on the eight disk array the measurements showed that as the size of the partial stripe increases the mean response time decreases, whereas in the analytical model [9] mean response time increases.

In order to improve the analytical model we analysed the physical behaviour on the disk when a large partial stripe write follows a full stripe write. Specifically, we focused on the amount of seeking each disk must do between the time that a partial stripe parity pre-read completes and the partial stripe write begins. The fewer disks that are pre-read ($n - b_{mod} - 1$), the more likely that the pre-read will complete before the remaining $b_{mod} + 1$ disks complete their respective full stripe writes. If any of the $b_{mod} + 1$ disks complete before the pre-read has completed servicing then that disk must wait to write the new data or parity. In this time, that disk may start servicing the next request in its queue, or just rotate away from the desired position. Henceforth, when the pre-read eventually completes, those disks will have to re-seek back causing additional seek and rotational latency. However, if the pre-read completes first then, when another disk completes its full stripe write, it can immediately write the new data or parity for the large partial stripe write without any additional seeking. We accordingly approximate the probability of the $b_{mod} + 1$ disks having to seek as $\frac{n-b_{mod}-1}{n}$.

However, as the number of full stripes written increases this relationship becomes less relevant. This is because each disk will take different amounts of time to write the (larger amount of) full stripe data and the additional pre-read time on some disks will be insignificant in comparison. The effect of zoning amplifies these differences. As the number of full stripes (k) increases, the disk that finishes first is less likely to depend on whether there was an additional pre-read on that disk, and it is more likely that all the disks will need to re-seek. Therefore, we define the probability of seeking as $1 - \frac{b_{mod}-1}{nk}$. Since all disks have to seek initially for the start of the full-stripe write, the mean seek time becomes $(1 - \frac{b_{mod}-1}{2nk}) (E[R] + E[S])$. All other parameters in the model remain the same as the previous model, so the cdf or request response time is:

$$W_{write}(t) = \left(W_d \left(\frac{t}{2}, \frac{\lambda(n + b_{mod} + 1)}{n} \right), \left(1 - \frac{b_{mod} - 1}{2nk} \right) (E[R] + E[S]) + E[T_{\frac{k+1}{2}}] \right)^{\frac{n+b_{mod}+1}{2}}$$

4 Validation

We demonstrate the accuracy of our models by validating them against a real eight disk RAID system. This is an improvement over [9], where RAID 01 was only validated for one size of request (2 blocks) on a four disk array, and also over [8] where both our RAID 01 and RAID 5 models were only validated against a four disk system.

Our experimental platform consists of an Infortrend A16F-G2430 RAID system containing eight Seagate ST3500630NS disks. Each disk has 60801 cylinders. A sector is 512 bytes and we have approximated, based on measurements from the disk drive, that the time to write a single physical sector on the innermost and outermost tracks are 0.012064ms (t_{max}) and 0.005976ms (t_{min}) respectively. The stripe width on the array is configured as 128KB, which we define as the block size. Therefore there are

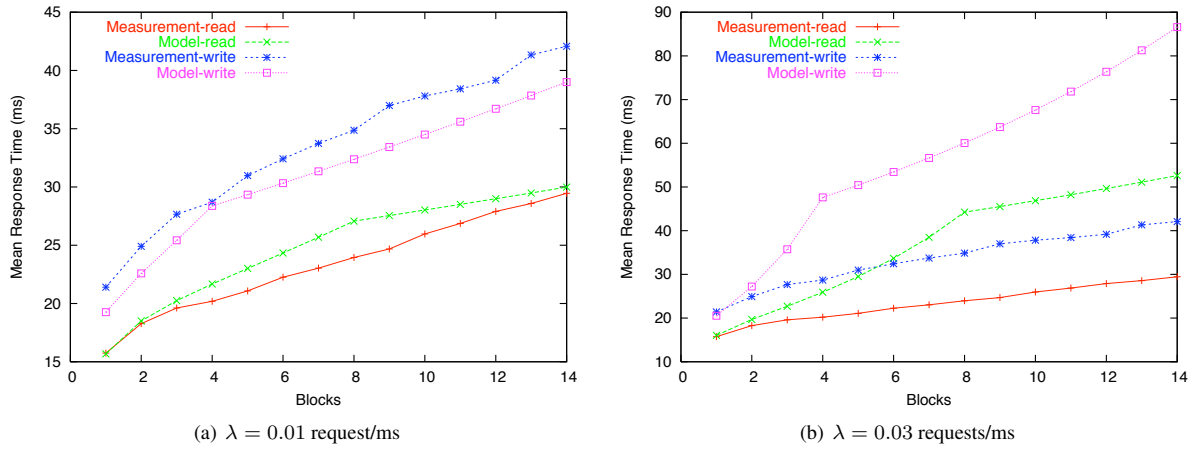


Figure 2: Comparison of mean response time against block size for RAID 01 for different values of λ .

256 sectors per block. The time for a full disk revolution is 8.33ms. A track to track seek takes 0.8ms and a full-stroke seek requires 17ms for a read request; the same measurements are 1ms and 18ms respectively for write requests [13].

To obtain response time measurements from this system, we implemented a benchmarking program that issues read and write requests using a master process and a number of child processes. These child processes are responsible for issuing and timing I/O requests, leaving the master free to spawn further processes without the need for it to wait for previously-issued operations to complete.

In order to validate the analytical model effectively, it was necessary to minimise the effects of buffering and caching as these are not currently represented in the model. We therefore disabled the RAID system's write-back cache, set the read-ahead buffer to 0 and opened the device with the `O_DIRECT` flag set. For each of the experiments presented below, 100 000 requests were issued and the resulting means, variances, pdfs and cdfs of the response times were calculated using the statistical package R.

4.1 RAID 01

Fig. 2 shows measured and modelled mean response times of reads and writes for RAID 01 for two different values of λ – a light load of $\lambda = 0.01$ requests/ms (Fig. 2(a)) and a heavy load of $\lambda = 0.03$ requests/ms (Fig. 2(b)). For requests under a light load, agreement between model and measurement is excellent. However under heavy load the model tends to increasingly overestimate. The RAID controller re-orders jobs in a queue for optimal performance [2], so a longer queue enables more re-ordering. This is not represented in our model yet, hence the disparity between model and measurement.

Fig. 3 compares pdfs and cdfs for some randomly chosen parameters. Interestingly, even if the measured and modelled cdfs do not have excellent agreement, their pdfs show some similar trends.

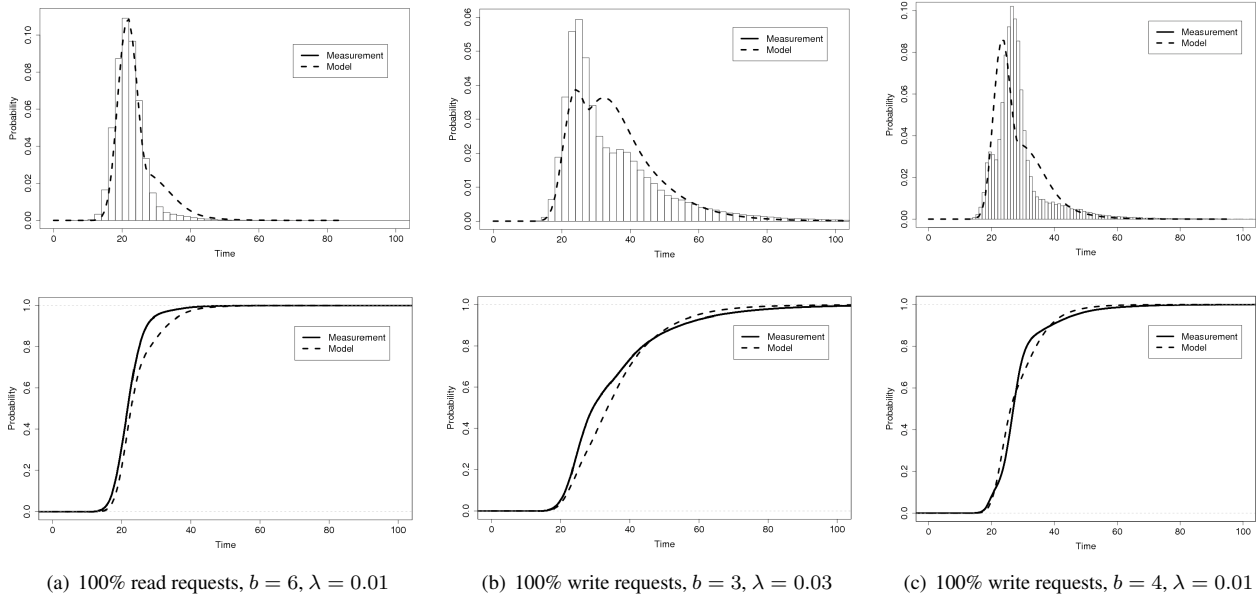


Figure 3: RAID 01 b -block request response time pdfs and cdfs for arrival streams of reads or writes with rate λ requests/ms.

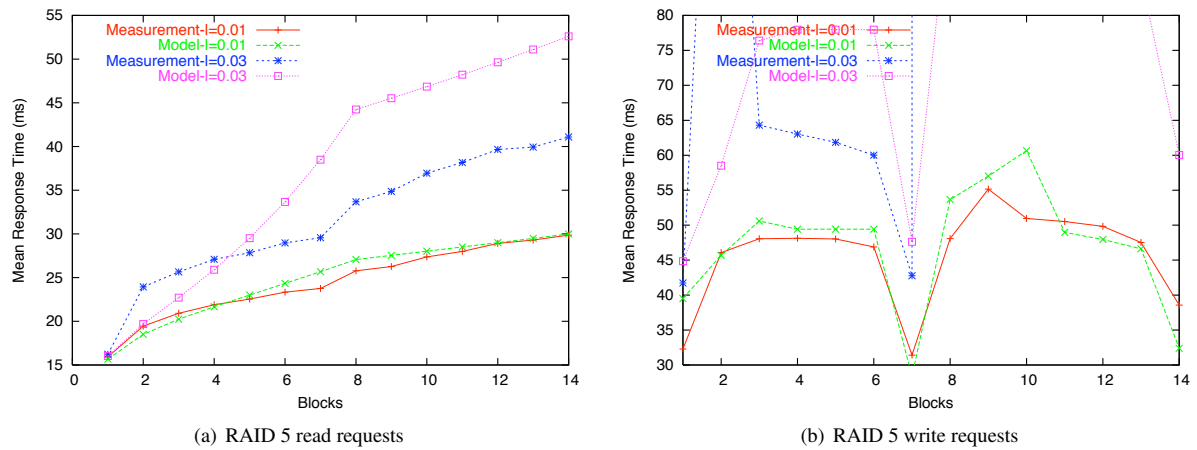


Figure 4: Comparison of mean response time against block size for RAID 5 for different values of λ .

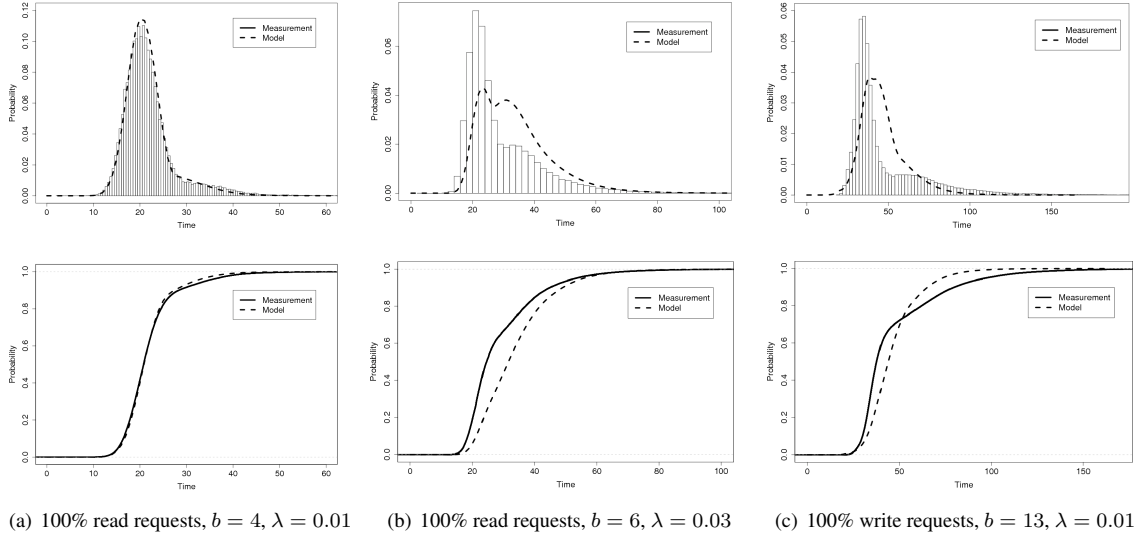


Figure 5: RAID 5 b -block request response time pdfs and cdfs for arrival streams of reads or writes with rate λ requests/ms.

4.2 RAID 5

Fig. 4(a) shows measured and modelled mean response times for RAID 5 reads under light and heavy loads. Similar to its RAID 01 equivalents, agreement is excellent for light load, but under heavier load for larger block sizes, the model increasingly overestimates the measurements. Fig. 4(b) shows measured and modelled results for RAID 5 writes under light and heavy loads. The dips for both measurement and model at 7 and 14 blocks occur because these are full stripe writes with no slow parity pre-reads. For light load there is good agreement between model and measurement. For a heavier load, both measurement and model quickly show signs of saturation.

Fig. 5 compares pdfs and cdfs for some randomly chosen parameters. The modelled pdf in Fig. 5(b) displays the bimodal nature of the measured result, but not the peak of the maximum value.

4.3 Mixed Reads and Writes

Fig. 6 shows measured and modelled mean response times for arrival streams with varying proportions of reads and writes for RAID 01 for two different values of λ (0.01 and 0.03), while Fig. 8 shows the same for RAID 5. In both cases, we again observe good agreement between measured and modelled results. Fig. 7 displays a selection of full pdf and cdf results for RAID 01 mixed reads and writes, while Fig. 9 contains a further selection for RAID 5. Particularly noteworthy is Fig. 9(a), in which the model accurately captures the bimodal distribution of the measured results.

We were particularly interested in determining if some apparently spurious measurement results in [8] could be reproduced on the eight disk array. For RAID 5 mixed reads and writes we observed extremely long mean response times for 2-blocks requests for all three mixes (25% reads, 50% reads and 75% reads) which were much

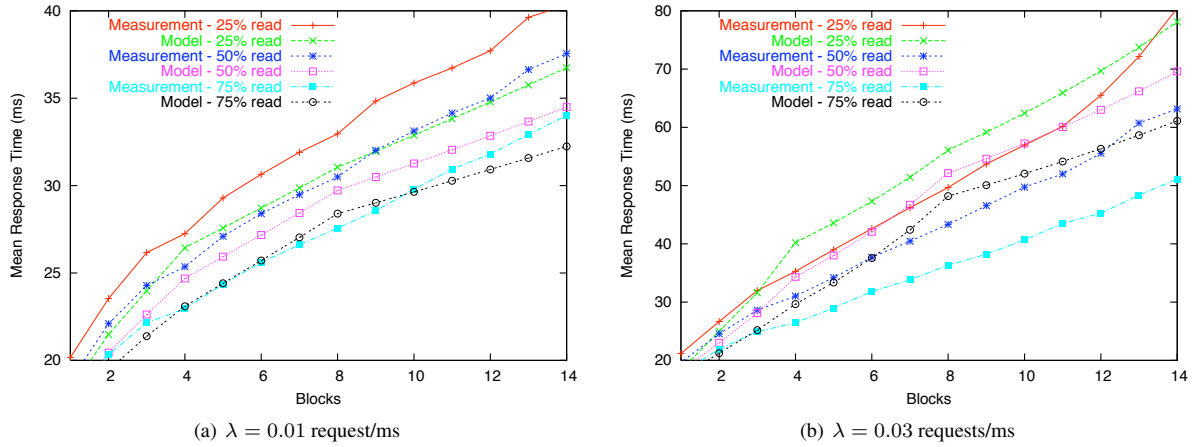


Figure 6: Comparison of mean response time against block size for RAID 01 for mixed arrival streams with different values of λ .

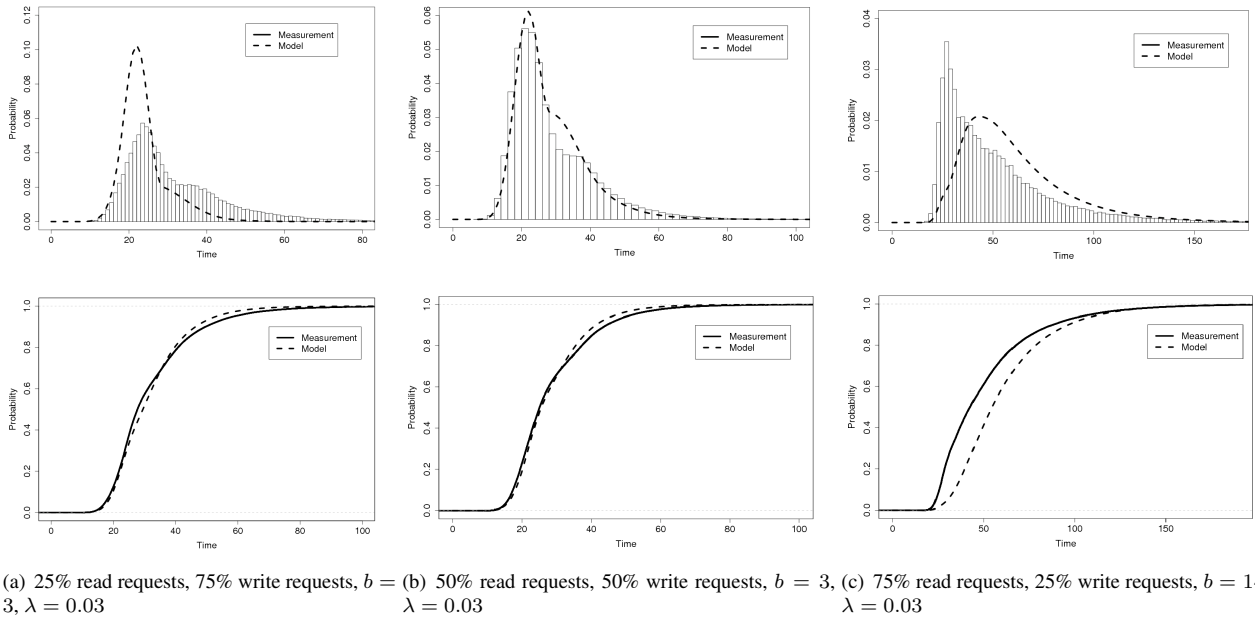


Figure 7: RAID 01 b -block request response time pdfs and cdfs for arrival streams of mixed reads and writes with rate λ requests/ms.

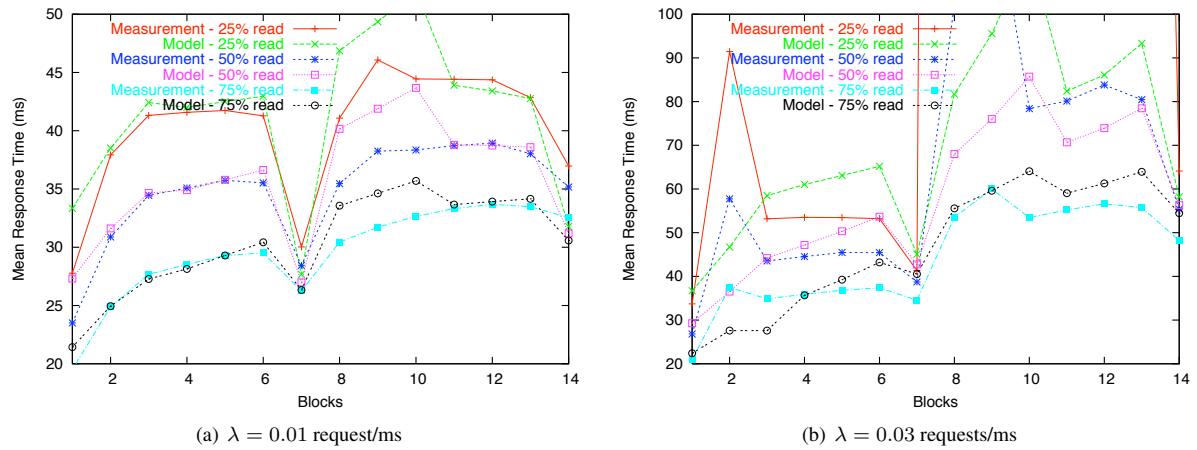


Figure 8: Comparison of mean response time against block size for RAID 5 for mixed arrival streams with different values of λ .

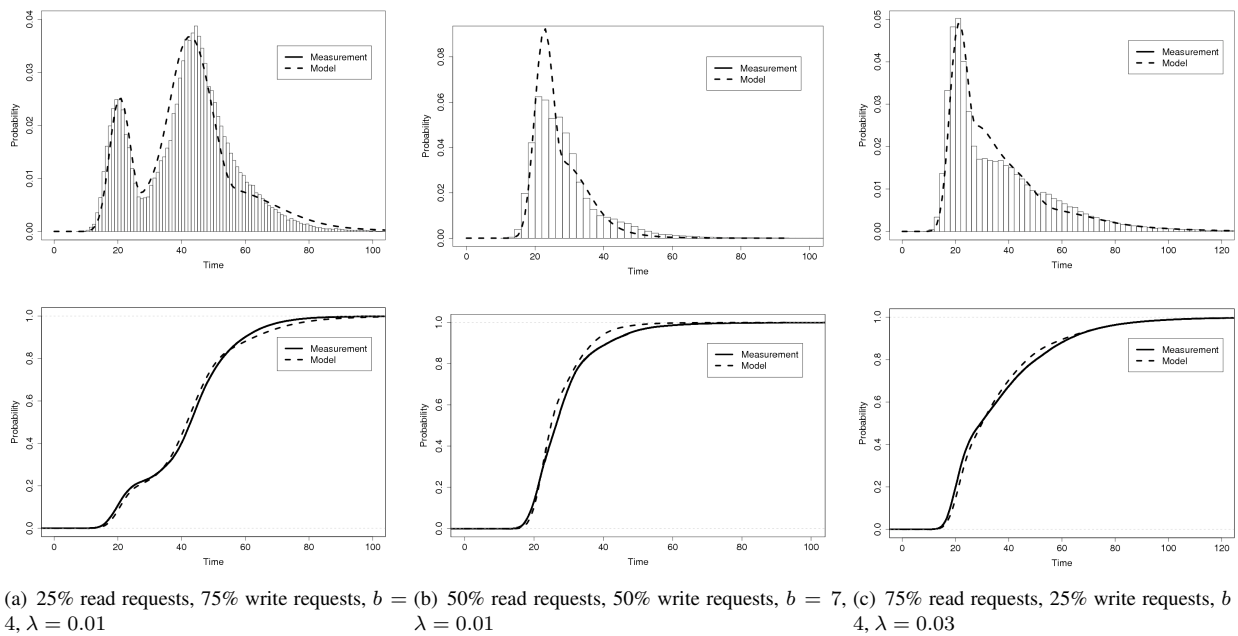


Figure 9: RAID 5 b -block request response time pdfs and cdfs for arrival streams of mixed reads and writes with rate λ requests/ms.

larger than the times for 2-block reads or 2-block writes and were not predicted by the model. In Table 4 we again observe this phenomenon, and indeed see that it is even more pronounced for 8- and 9-block transfers at $\lambda = 0.03$, suggesting it was not just an artifact of the four disk configuration. Further investigation suggests that it is a result of constraining all operations to begin at the start of a stripe, leading to unequal load on some disks, as when this restriction is relaxed it does not occur.

5 Conclusion

In this paper we have presented an improved performance model for RAID systems capable of calculating full request response time distributions. In particular, we have improved the RAID 5 large partial stripe following full stripe write model to more closely accurately observed behaviour. We validated our models for RAID 01 and 5 for reads, writes and mixtures of the two on a real-life RAID array with eight disks.

There are a number features which we still need to model in order to have a comprehensive model capable of representing real I/O workloads. Firstly, caching is not yet supported in our model. Secondly, we would like to support sequential as well as random I/O, to better model the effects of locality. Thirdly, we currently constrain the alignment of RAID 5 write requests to start at the beginning of a stripe in all cases. In the future, we would like to allow for requests that start with a partial stripe, followed by further data. Preliminary investigations suggest that this also remedies some of the unusual measurements for mixed reads and writes on RAID 5. Fourthly, all our models assume fixed request sizes and we would like to extend them to incorporate distributions of block sizes. Fifthly, we need to model the effect of the re-ordering of requests by the RAID array when greater load is experienced. Finally, we have assumed Markovian arrivals in our model, and have generated request streams that conform to this assumption for our measurements. We intend to compare the model response times with response times generated from real I/O traces.

Acknowledgements

We are grateful to Peter Harrison and Soraya Zertal for helpful discussions. This work is supported by EPSRC research grant “Intelligent Performance Optimisation of Virtualised Data Storage Systems” (iPODS) (EP/F010192/1), with industrial partners IBM and Reuters.

References

- [1] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems Theory and Applications*, 10(1-2):5–88, 1992.
- [2] D. Anderson. You don’t know jack about disks. *Queue*, 1(4):20–30, 2003.
- [3] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.
- [4] S. Chen and D. Towsley. A performance evaluation of RAID architectures. *IEEE Transactions on Computers*, 45(10):1116–1130, 1996.

- [5] H. A. David. *Order Statistics*. John Wiley and Sons, Inc, 1981.
- [6] P. G. Harrison and N. M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1993.
- [7] P. G. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation*, 64(7-8):664–689, August 2007.
- [8] A. S. Lebrecht, N. J. Dingle, and W. J. Knottenbelt. Modelling and validation of response times in zoned RAID. In *16th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '08)*, September 2008. To appear.
- [9] A. S. Lebrecht, N. J. Dingle, and W. J. Knottenbelt. A response time distribution model for zoned RAID. In *15th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA)*, June 2008.
- [10] A. S. Lebrecht and W. J. Knottenbelt. Response time approximations in fork-join queues. In *23rd UK Performance Engineering Workshop (UKPEW)*, July 2007.
- [11] E. K. Lee. *Performance Modeling and Analysis of Disk Arrays*. PhD thesis, University of California at Berkeley, 1993.
- [12] R. Nelson and A. N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Transactions on Computers*, 37(6):739–743, June 1988.
- [13] Seagate. Barracuda ES Data Sheet, 2007. <http://www.seagate.com/docs/pdf/datasheet/disc/ds.barracuda.es.pdf>.
- [14] A. Thomasian and A. N. Tantawi. Approximate solutions for M/G/1 fork/join synchronization. In *Proc. WSC '94*, pages 361–368, 1994.
- [15] E. Varki. Mean value technique for closed fork-join networks. In *Proc. ACM SIGMETRICS*, pages 103–112, 1999.
- [16] E. Varki. Response time analysis of parallel computer and storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1146–1161, 2001.
- [17] E. Varki, A. Merchant, J. Xu, and X. Qiu. Issues and challenges in the performance analysis of real disk arrays. *IEEE Transactions on Parallel and Distributed Systems*, 15(6):559–574, June 2004.
- [18] S. Varma and A. M. Makowski. Interpolation approximations for symmetric fork-join queues. In *Proc. Performance '93*, pages 245–265, 1994.
- [19] S. Zertal and P. G. Harrison. Multi-RAID queueing model with zoned disks. In *High Performance Computing and Simulation Conference (HPCS'07)*, 2007.

Appendix

λ (ms ⁻¹)	# Blks	Reads				Writes			
		Measured		Modelled		Measured		Modelled	
		Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)
0.01	1	15.7	15.4	15.7	20.8	21.4	26.8	19.3	19.6
	2	18.3	13.0	18.5	18.0	24.9	38.8	22.6	27.3
	3	19.6	13.1	20.2	20.2	27.7	60.7	25.4	41.6
	4	20.2	15.2	21.7	24.6	28.7	77.2	28.4	58.4
	5	21.1	16.9	23.0	30.5	31.0	89.4	29.3	63.5
	6	22.3	21.4	24.3	37.2	32.4	102.5	30.3	68.9
	7	23.0	22.9	25.7	44.5	33.7	118.1	31.3	74.8
	8	24.0	27.7	27.1	52.0	34.9	132.5	32.4	80.9
	9	24.7	29.4	27.5	54.4	37.0	155.0	33.4	87.5
	10	26.0	34.4	28.0	56.8	37.8	164.2	34.5	94.4
	11	26.9	38.4	28.5	59.3	38.4	148.0	35.6	101.7
	12	27.9	42.1	29.0	61.8	39.2	147.3	36.7	109.5
	13	28.6	45.1	29.5	64.5	41.3	170.4	37.9	117.7
	14	29.5	50.0	30.0	67.2	42.1	173.5	39.0	126.3
0.03	1	15.7	15.4	16.0	25.2	21.4	26.8	20.5	35.2
	2	18.3	13.0	19.7	31.7	24.9	38.8	27.2	80.0
	3	19.6	13.1	22.7	47.9	27.7	60.7	35.7	154.6
	4	20.2	15.2	25.9	70.6	28.7	77.2	47.6	283.1
	5	21.1	16.9	29.5	99.1	31.0	89.4	50.4	322.1
	6	22.3	21.4	33.7	134.2	32.4	102.5	53.4	366.9
	7	23.0	22.9	38.5	179.0	33.7	118.1	56.6	418.1
	8	24.0	27.7	44.2	239.2	34.9	132.5	60.1	476.9
	9	24.7	29.4	45.5	255.1	37.0	155.0	63.7	544.6
	10	26.0	34.4	46.9	272.3	37.8	164.2	67.6	622.6
	11	26.0	38.4	48.2	290.5	38.4	148.0	71.8	712.9
	12	27.9	42.1	49.6	310.1	39.2	147.3	76.4	817.4
	13	28.6	45.0	51.1	331.0	41.3	170.4	81.3	939.2
	14	29.5	50.0	52.6	353.3	42.1	173.5	86.6	1081.8

Table 1: Mean and variance of request response time for RAID 01 reads and writes against measured results.

λ (ms ⁻¹)	# Blks	Reads				Writes			
		Measured		Modelled		Measured		Modelled	
		Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)
0.01	1	15.9	17.0	15.7	20.8	32.3	129.2	39.5	192.0
	2	19.5	32.4	18.5	18.0	46.1	566.2	45.7	193.8
	3	20.9	34.2	20.2	20.2	48.1	132.0	50.6	232.5
	4	21.9	34.1	21.7	24.6	48.1	128.4	49.4	201.6
	5	22.6	34.3	23.0	30.5	48.0	124.4	49.4	201.6
	6	23.3	35.1	24.3	37.2	46.9	118.9	49.4	201.6
	7	23.8	35.6	25.7	44.5	31.4	83.7	28.4	58.4
	8	25.8	52.7	27.1	52.0	48.1	1042.2	53.7	269.6
	9	26.3	56.2	27.5	54.4	55.2	1464.3	57.0	312.6
	10	27.4	64.8	28.0	56.8	51.0	699.5	60.7	362.8
	11	28.0	68.8	28.5	59.2	50.5	699.7	49.0	213.1
	12	28.9	73.7	29.0	61.8	49.8	668.5	48.0	204.6
	13	29.3	73.9	29.5	64.5	47.5	590.5	46.6	193.6
	14	29.9	78.5	30.0	67.2	38.6	121.8	32.4	80.9
0.03	1	16.2	21.5	16.0	25.2	41.7	408.1	44.9	346.8
	2	23.9	106.7	19.7	31.7	152.6	11040.6	58.5	573.4
	3	25.7	116.6	22.7	47.9	64.3	539.6	76.4	1036.4
	4	27.1	129.2	25.9	70.6	63.0	524.6	78.0	1052.9
	5	27.9	135.0	29.5	99.0	61.9	517.8	78.0	1052.9
	6	29.0	148.2	33.7	134.2	60.0	537.7	78.0	1052.9
	7	29.6	153.8	38.5	179.0	42.8	516.9	47.6	283.1
	8	33.7	196.8	44.2	239.1	sat	sat	96.9	1701.3
	9	34.9	225.3	45.5	255.1	sat	sat	119.3	2628.9
	10	37.0	247.4	46.9	272.2	sat	sat	152.9	4399.6
	11	38.2	281.1	48.2	290.5	sat	sat	91.2	1388.5
	12	39.7	295.3	49.6	310.0	sat	sat	89.2	1300.0
	13	39.9	298.0	51.1	330.9	sat	sat	85.4	1167.2
	14	41.1	321.6	52.6	353.2	sat	sat	60.1	476.8

Table 2: Mean and variance of request response time for RAID 5 reads and writes against measured results.

λ (ms ⁻¹)	# Blks	25% Reads, 75% Writes				50% Reads, 50% Writes				75% Reads, 25% Writes			
		Measured		Modelled		Measured		Modelled		Measured		Modelled	
		Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)
0.01	1	20.2	30.2	18.3	21.8	18.8	30.5	17.4	22.6	17.3	25.4	16.6	22.3
	2	23.5	41.0	21.5	26.4	22.1	40.6	20.4	24.4	20.3	32.0	19.5	21.6
	3	26.2	60.6	24.0	37.8	24.3	55.6	22.6	32.6	22.2	41.4	21.4	26.5
	4	27.2	73.5	26.4	52.1	25.4	66.0	24.7	43.7	23.0	49.2	23.1	34.3
	5	29.3	90.2	27.6	57.6	27.1	82.9	25.9	49.7	24.4	61.0	24.4	40.4
	6	30.7	104.5	28.7	64.0	28.4	95.4	27.2	56.7	25.6	71.0	25.7	47.6
	7	31.9	120.3	29.9	71.1	29.5	106.5	28.4	64.6	26.6	78.0	27.0	55.7
	8	33.0	130.7	31.1	79.0	30.5	115.7	29.7	73.5	27.6	86.6	28.4	64.5
	9	34.8	154.1	32.0	85.7	32.0	137.3	30.5	79.6	28.6	96.1	29.0	69.1
	10	35.9	163.6	32.9	92.9	33.1	143.0	31.3	86.1	29.8	103.8	29.6	74.1
	11	36.7	156.1	33.8	100.6	34.2	147.7	32.1	93.1	30.9	113.6	30.3	79.3
	12	37.7	164.1	34.8	108.7	35.0	151.2	32.9	100.6	31.8	114.9	30.9	84.9
	13	39.6	186.2	35.8	117.5	36.6	176.9	33.7	108.6	32.9	132.5	31.6	90.9
	14	40.3	187.3	36.8	126.8	37.6	181.3	34.5	117.1	34.0	141.0	32.2	97.2
0.03	1	21.2	49.1	18.3	21.8	19.7	46.4	18.2	32.3	17.8	35.1	17.1	29.2
	2	26.7	101.8	25.0	68.7	24.6	85.4	23.0	56.2	21.9	61.0	21.2	43.6
	3	32.0	189.6	31.6	126.8	28.6	140.7	28.1	98.4	24.9	91.2	25.2	71.6
	4	35.3	301.2	40.2	217.2	31.0	199.2	34.4	159.4	26.5	118.0	29.7	110.7
	5	39.0	405.3	43.6	256.0	34.2	269.1	38.0	197.0	29.0	160.0	33.4	144.2
	6	42.6	529.8	47.3	308.1	37.7	372.2	42.1	247.3	31.8	222.8	37.6	188.6
	7	46.2	671.2	51.4	375.1	40.5	458.3	46.7	317.5	33.9	273.3	42.4	250.8
	8	49.7	836.0	56.1	464.3	43.3	592.7	52.1	420.5	36.3	341.9	48.2	345.5
	9	53.7	1012.3	59.2	534.2	46.6	685.4	54.6	482.5	38.2	388.8	50.1	389.5
	10	57.0	1240.0	62.4	615.9	49.7	812.5	57.2	555.3	40.7	451.2	52.1	440.7
	11	60.2	1491.2	65.9	711.8	52.0	944.3	60.0	641.1	43.5	551.2	54.1	500.6
	12	65.5	2472.3	69.7	824.6	55.5	1212.1	63.0	742.4	45.3	615.0	56.3	570.9
	13	72.1	3206.3	73.7	957.9	60.7	1739.7	66.2	862.8	48.4	788.1	58.6	653.8
	14	80.4	5305.8	78.1	1116.3	63.2	1824.8	69.6	1006.3	51.1	896.7	61.1	752.0

Table 3: Comparison of mean response times and variances for mixed read and write request streams for RAID 01.

λ (ms ⁻¹)	# Blks	25% Reads, 75% Writes				50% Reads, 50% Writes				75% Reads, 25% Writes			
		Measured		Modelled		Measured		Modelled		Measured		Modelled	
		Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)	Mean (ms)	σ^2 (ms ²)
0.01	1	27.8	130.8	33.3	241.9	23.5	109.1	27.3	227.1	19.5	68.2	21.4	152.1
	2	37.9	453.6	38.5	260.2	30.9	312.4	31.6	247.9	24.9	173.2	25.0	164.8
	3	41.3	233.8	42.4	304.0	34.5	258.4	34.7	284.1	27.7	189.9	27.3	185.7
	4	41.6	227.5	42.0	261.2	35.1	250.3	34.9	244.9	28.5	185.9	28.1	163.0
	5	41.7	222.1	42.5	258.5	35.7	247.1	35.8	244.5	29.3	189.4	29.3	166.5
	6	41.3	205.6	43.0	257.4	35.5	227.7	36.6	246.0	29.5	174.1	30.4	171.3
	7	30.0	92.2	27.7	55.4	28.5	94.3	27.0	52.1	26.3	73.5	26.3	48.5
	8	41.1	641.9	46.9	323.4	35.5	398.6	40.2	302.9	30.4	216.6	33.6	211.4
	9	46.1	961.2	49.4	368.0	38.3	576.4	41.9	339.1	31.7	269.7	34.6	232.6
	10	44.5	569.8	52.0	418.9	38.4	419.2	43.7	379.2	32.7	241.4	35.7	255.2
	11	44.4	562.5	43.9	214.7	38.7	415.2	38.8	191.1	33.3	253.0	33.7	139.9
	12	44.4	547.0	43.4	196.8	38.9	407.7	38.7	173.3	33.7	242.7	33.9	129.7
	13	42.8	474.3	42.8	179.1	38.0	351.5	38.6	156.6	33.5	216.6	34.2	120.3
	14	37.0	145.3	31.8	78.6	35.2	150.9	31.2	75.5	32.6	126.5	30.6	71.7
0.03	1	33.7	337.2	36.7	355.6	26.8	235.0	29.3	295.7	21.0	123.3	22.4	181.8
	2	91.5	4961.0	46.7	535.7	57.8	2138.8	36.5	414.7	37.5	804.9	27.6	240.3
	3	53.2	597.0	58.5	863.5	43.5	545.4	44.2	612.8	34.9	395.3	27.6	240.3
	4	53.5	604.0	61.0	888.4	44.5	551.1	47.2	641.3	35.9	397.1	35.7	360.1
	5	53.5	603.2	63.1	920.7	45.5	575.8	50.3	693.2	36.8	407.4	39.2	411.0
	6	53.2	621.4	65.2	960.8	45.4	560.3	53.7	757.3	37.4	402.0	43.2	474.1
	7	41.3	479.6	45.1	256.2	38.7	411.1	42.7	229.7	34.6	301.3	40.5	203.9
	8	1341.1	695702	81.7	1467.7	102.3	10900	68.0	1129.0	53.5	1884.4	55.6	712.7
	9	2250.1	2177073	95.6	2057.0	130.2	16333	76.0	1456.8	60.0	2562.2	59.6	851.7
	10	170.2	60233	114.0	2993.3	78.4	3753.9	85.7	1907.0	53.5	1357.5	64.1	1020.1
	11	193.7	74681	82.4	1077.8	80.1	4047.0	70.7	743.3	55.2	1456.8	59.0	496.3
	12	350.1	186247	86.1	1390.2	83.8	5213.8	73.9	833.3	56.6	1587.7	61.29	517.8
	13	444.7	211185	93.3	2669.0	80.5	4935.4	78.5	1116.3	55.7	1511.8	63.9	578.3
	14	64.1	3934.3	58.2	456.3	55.4	1182.5	56.3	428.9	48.2	677.6	54.5	394.6

Table 4: Comparison of mean response times and variances for mixed read and write request streams for RAID 5.

Response time distributions via reversed processes

Peter G. Harrison*

Maria G. Vigliotti*

Abstract

Response time calculations in stochastic networks – e.g. queueing networks – are usually developed in terms of sample path analyses beginning in an equilibrium state. We consider the joint probability distribution of the sojourn times of a tagged task at each node in a network and observe that this is the same in both the forward and reversed processes. Therefore if the reversed process is known, each node-sojourn time can be taken from either process. In particular, the reversed process can be used for the last node in a path and the forward process for the other nodes in a recursive analysis. In this way we can derive quickly and systematically existing results for response time probability densities in tandem, open and closed tree-like, and overtake-free Markovian networks of queues. We also show how to apply the method in non-queueing systems.

1 Introduction

The response time of a particular, ‘tagged’ task along a path in a network of nodes of some kind is defined to be the sum of the sojourn times of the task (i.e. its delays) at those nodes that constitute the path. More generally, the response time distribution follows directly from the joint probability distribution of the node-sojourn times. For a path comprising the sequence of nodes $(1, 2, \dots, m)$, let the response time $R = T_1 + T_2 + \dots + T_m$, where T_i is the sojourn time at node i , $(1 \leq i \leq m)$, with probability distribution function $T_i(t)$. Then the joint sojourn time distribution is $J(t_1, \dots, t_m) = \mathbb{P}(T_1 \leq t_1, \dots, T_m \leq t_m)$ and, denoting Laplace-Stieltjes transforms (LSTs) by asterisks, the m -dimensional LST of the joint sojourn time distribution is

$$J^*(\theta_1, \dots, \theta_m) = \int_0^\infty \dots \int_0^\infty e^{-(\theta_1 t_1 + \dots + \theta_m t_m)} \mathbf{d}J(t_1, \dots, t_m)$$

The response time distribution then has LST $R^*(\theta) = J^*(\theta, \dots, \theta)$. When the sojourn times T_i are independent, this simplifies to $R^*(\theta) = \prod_{i=1}^m T_i^*(\theta)$.

If the sojourn time at each node i depends solely on the state, N_i say, existing at the node *immediately prior to the arrival of the tagged task*, the conditional joint sojourn time LST is $J^*(\theta_1, \dots, \theta_m \mid \mathbf{n}) = \prod_{i=1}^m T_i^*(\theta_i \mid n_i)$ where $T_i^*(\theta_i \mid n_i) = \int_0^\infty e^{-\theta_i t} \mathbf{d}\mathbb{P}(T_i \leq t \mid N_i = n_i)$ ¹. In such networks, response

*Department of Computing, Imperial College London, {pgh,mgv98}@doc.ic.ac.uk

¹For example, when $m = 2$, $J^*(\theta_1, \theta_2 \mid \mathbf{N} = \mathbf{n}) = \mathbb{E}[\mathbb{E}[e^{-(\theta_1 T_1 + \theta_2 T_2)} \mid T_1, \mathbf{N} = \mathbf{n}] \mid \mathbf{N} = \mathbf{n}] = \mathbb{E}[e^{-\theta_1 T_1} \mathbb{E}[e^{-\theta_2 T_2} \mid T_1, \mathbf{N} = \mathbf{n}] \mid \mathbf{N} = \mathbf{n}] = \mathbb{E}[e^{-\theta_1 T_1} \mathbb{E}[e^{-\theta_2 T_2} \mid N_2 = n_2] \mid N_1 = n_1]$.

time distributions can be computed iteratively through their LSTs using the result that:

$$J^*(\theta_1, \dots, \theta_m \mid \mathbf{1}) = \Pi_{i=1}^m T_i^*(\theta_i \mid n_i) \mathbb{P}(\mathbf{N} = \mathbf{n} \mid \mathbf{L}(0) = \mathbf{1})$$

where bold type indicates vectors and the random variable $L_i(t)$ is the state of node i at time t , so that the initial state is $\mathbf{L}(0)$ and $N_i = L_i(T_i^-)$ when the tagged task arrives at node i at time T_i . Of course, if the N_i are independent for $i = 1, \dots, m$, this reduces to the above result that $J^*(\theta_1, \dots, \theta_m) = \Pi_{i=1}^m T_i^*(\theta_i)$.

In queueing networks it is often the case that the node sojourn times depend only on the queue length at the arrival instant, for example in the overtake-free networks of [9], but the computation of the transient probabilities $\mathbb{P}(\mathbf{N} = \mathbf{n} \mid \mathbf{L}(0) = \mathbf{1})$ is problematic; see [5] for example. If these probabilities can be found (or avoided), the method applies in both open and closed networks; see the above citations, for example.

In the present contribution, we apply a completely different approach to the computation of the LSTs of response times in Markovian networks at equilibrium, via joint sojourn time distributions, using reversed processes. The idea is based on the observation that sojourn times are the same whether one considers the forward process or its reversed process. When sojourn times depend only on the state existing at a node at the arrival instant and the reversed process is separable, i.e. a synchronising network of m reversed nodes, we can use the forward sojourn time at node 1 and the reversed sojourn time at the last node m ; a recursive analysis allows us to consider only the case $m = 2$, nodes $1, \dots, m - 1$ constituting a single aggregate ‘super-node’ in the recursion.

In the next section, we define our method and apply it to a range of queueing networks, providing greatly simplified derivations that hold the potential of automation through the reversed compound agent theorem (RCAT). Simple properties of response times in G-networks, which are actually non-queueing networks with very different response time characteristics [6], follow immediately, and an alternative methodology is revealed in more complex cases. Generalisations are considered in section 4, focusing on a tandem pair of first-come-first-served (FCFS) queues with Erlangian service times; note that in general, such networks do not even possess a product-form solution. The paper concludes in section 5 where future potential of the method is evaluated.

2 Node-sojourn times and reversed processes

First, let us consider the sojourn times spent by a task in a pair of nodes that are connected in the sense that the task first sojourns in node 1, for time T_1 , after which it proceeds to node 2 and sojourns there, for time T_2 , before departing from the system. Suppose that the *initial state* of the system, i.e. that existing immediately prior to the arrival of the task at node 1, is \mathbf{s}_0 . In many cases, e.g. a pair of tandem queues, the state \mathbf{s} is a pair, $\mathbf{s} = (s_1, s_2)$, where s_i describes the state of node i only, $i = 1, 2$. We call such a state *separable*.

The sojourn time at node 1, T_1 say, can be calculated as the first passage time from the initial state to exit from the state in which the task departs node 1. In general, this can involve arbitrary transitions in the whole system, i.e. be influenced by the evolution of node 2 as well as node 1. However, often, T_1 is determined solely by the initial state and the evolution of node 1, as in

the case of constant rate queues, for example. In this case, the conventional approach to sojourn time analysis is to consider the state of the system at the instant of the task's departure from node 1 and use this as the initial state for the sojourn at node 2; this may also (or may not, of course) then depend solely on the evolution of node 2.

The properties we need to use this technique are therefore:

- The state of the system is separable, i.e. $\mathbf{s} = (s_1, s_2)$, where s_i describes the state of node i only, $i = 1, 2$;
- The sojourn time of the tagged task at each node depends *solely* on the node's state at its arrival instant – implying that the node has the ‘overtake-free’ property of [9] which requires that the passage of the tagged task through the node is not influenced by tasks at any other node;
- The sojourn time at each node can be characterised as a first passage time in a Markov chain describing the node's behaviour during that sojourn *insofar as it affects the tagged task*.

Notice that the last point does not necessarily require the Markov chain describing the whole system or even the node: for example a transient chain representing a queue with no arrivals is sufficient if the first property holds. This is a traditional approach that was used to obtain the Laplace transform of response time distributions in cyclic, tree-like and overtake-free networks in the 1980s [9, 2, 5].

Our alternative approach uses the observation that sojourn times are the same whether one considers the forward process or its reversed process. Thus, given initial state $\mathbf{s}_0 = (s_{0,1}, s_{0,2})$ in a two-node network, we may take the sojourn time at the first node in the forward process (conditioned on $s_{0,1}$) and the *reversed sojourn time* at the second node in the reversed process, conditioned on the state at the end of the two sojourns. Notice that the reversed sojourn time is not necessarily dependent on only the initial state pertaining to the second node (final state in the forwards process). Indeed, the reversed process itself may depend on the joint state of the whole system, even if the forward node was overtake-free.

Let the reversed sojourn time at node i be denoted by \tilde{T}_i . Then the LST of the joint sojourn time distribution can be written

$$\begin{aligned} J^*(\theta_1, \theta_2) &= \mathbb{E}[\mathbb{E}[e^{-(\theta_1 T_1 + \theta_2 \tilde{T}_2)} \mid \mathbf{S}(0)]] \\ &= \mathbb{E}[\mathbb{E}[\mathbb{E}[e^{-(\theta_1 T_1 + \theta_2 \tilde{T}_2)} \mid T_1, \mathbf{S}(0)] \mid \mathbf{S}(0)]] \\ &= \mathbb{E}[\mathbb{E}[e^{-\theta_1 T_1} \mathbb{E}[e^{-\theta_2 \tilde{T}_2} \mid T_1, \mathbf{S}(0)] \mid \mathbf{S}(0)]] \end{aligned} \quad (1)$$

where the random variable $\mathbf{S}(t)$ denotes the state of the system at time t . Now suppose that the state vector is separable, so that $\mathbf{S}(t) = (S_1(t), S_2(t))$ as described above, and that the reversed sojourn time at node 2 depends only on the state existing at the arrival instant of the tagged task in the reversed process. Then we have

$$J^*(\theta_1, \theta_2) = \mathbb{E}[\mathbb{E}[e^{-\theta_1 T_1} \mathbb{E}[e^{-\theta_2 \tilde{T}_2} \mid S_2(0)] \mid \mathbf{S}(0)]]$$

If further the (forward) sojourn time at node 1 depends only on its initial state S_1 , we find

$$J^*(\theta_1, \theta_2) = \mathbb{E}_{S_1, S_2}[T_1^*(\theta_1 \mid S_1(0)) \tilde{T}_2^*(\theta_2 \mid S_2(0))] \quad (2)$$

To summarise, the conditions we need to apply equation 2 to a two-node network are:

1. The state of the system separable;
2. The sojourn time at node 1 depends *solely* on the initial state at node 1;
3. The reversed sojourn time at node 2 depends *solely* on the initial state at node 2;
4. The sojourn, respectively reversed sojourn, time at nodes 1 and 2 can be characterised as first passage times in Markov chains describing the respective node's behaviour during that sojourn.

Conditions 3 and 4 are aided by a specification of the reversed process for node 2. This may be provided by the Reversed Compound Agent Theorem (RCAT), which induces a systematic way to construct the reversed process of a separable synchronisation between two Markov chains [3]; see the next two sections. Note that the reversed response time in the second node is not, in general, a response time in the same sense and may be hard to determine even if the reversed process of the node is known, e.g. by RCAT. The above conditions can be relaxed, according to equation 1, but the ensuing analysis is very much more complex, involving the evolution of the joint state.

Paths of more than two nodes can be handled recursively, building a path by adding one node at a time – at each stage, a two-node path is considered comprising the current (partial) path as one node and the new node added as the other. This method is powerful and derives all the known results on response time distributions in overtake-free queueing networks, as we discuss in section 3. Furthermore, it could open the door to non-queueing applications, but it must be remembered that the above conditions are quite stringent, especially the third and fourth.

In the next section we illustrate the new technique in queueing networks, obtaining a concise explanation of several previous results. We then consider further applications in section 4, comparing the conditions required with the separable reversed process that may be given by RCAT.

3 Queueing networks

Queueing networks are relatively tractable since the M/M/1 queue is *reversible*, i.e. its reversed process is the same M/M/1 queue – a result routinely derivable by RCAT, but a well known fact anyway [8, 5, 3]. Moreover, the queue left behind by any departing task comprises precisely the tasks that arrived during its sojourn. Therefore, we have the following result:

Proposition 3.1 *At equilibrium, the reversed sojourn time in an M/M/1 queue has the same probability distribution as the forward sojourn time.*

Proof In the reversed process, the initial state is the number of tasks that arrived during the (forward) sojourn of the tagged task, n say. Consequently, the reversed sojourn time is the sum of $n + 1$ service times.² Conditioned on

²This uses the fact that the residual service time of the task being served on arrival (i.e. reversed departure) of the tagged task is distributed as a full (exponential) service time.

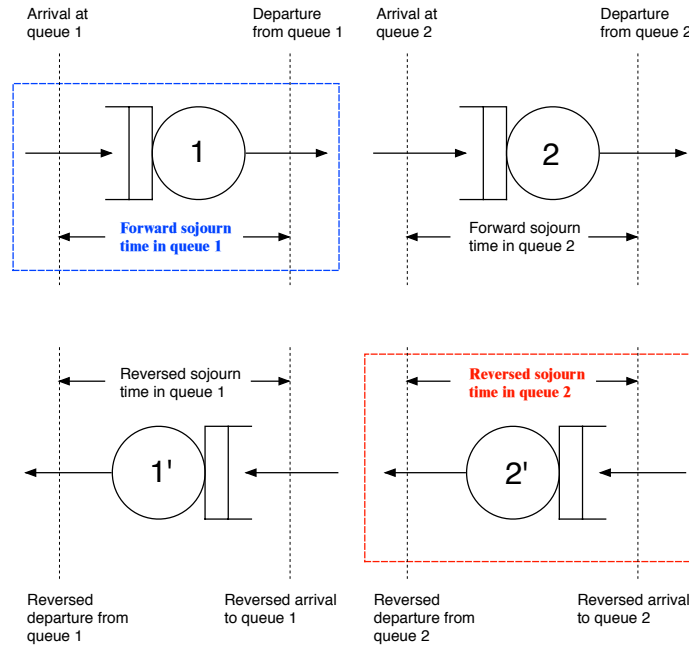


Figure 1: Two M/M/1 queues in tandem and the reversed process

their respective initial states, therefore, the reversed sojourn time is equal in distribution to the (forward) sojourn time. The result now follows since the equilibrium probability distribution of the queue length immediately before an arrival is the same in both processes, these both being an M/M/1 queue. ♠
The result of the previous section is now easy to apply, in both open and closed

queueing networks. We begin with a tandem pair and a cycle of two M/M/1 queues.

3.1 Tandem and cyclic pair of queues

Consider first the tandem pair of queues depicted in figure 1 – the cyclic counterpart is simply obtained by connecting the departures of the second queue to the arrivals of the first.

The forward and reversed nodes are both shown, and the forward response time at node 1 and reversed response time at node 2 are illustrated, as per section 2. Possible sample paths for the forward node 1 and node 2 processes are shown in figure 2, which illustrate the passage of the tagged task through the network. This arrives at node 1 to find a queue of length 4 and, on departing from node 1, finds a queue of length 3 at node 2. The traditional method of analysis investigates such sample paths and needs to consider the (transient) probability distribution of the node 2 queue length at the departure instant of the tagged task from node 1.

In our alternative approach, we consider the sample paths in the forward and reversed processes together, beginning in the same initial state – (4,8) in

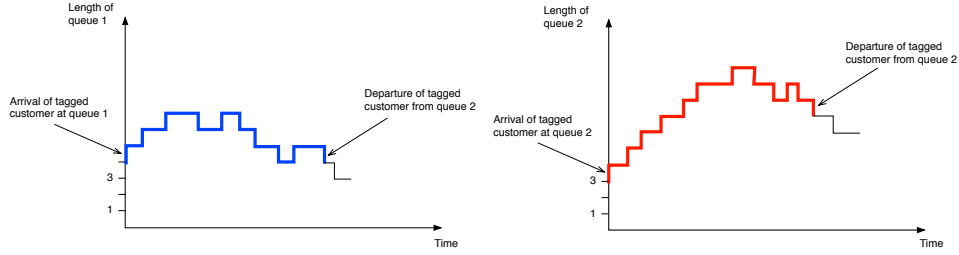


Figure 2: Possible sample paths for the queue lengths at each queue during the sojourn of the tagged task

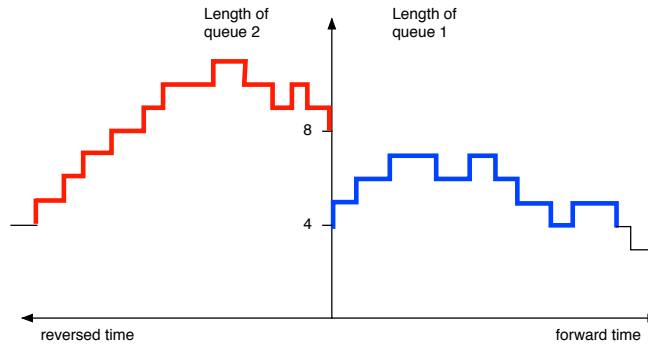


Figure 3: Forward and reversed sample paths given initial state (4,8)

the sample paths shown in figure 3. For the forward response time at node 1 we look to the right of the vertical axis and for the reversed response time at node 2 we look to the left. However, by equation 2, we only need to condition on the initial joint state. Since forward and reversed sojourn times are identically distributed by proposition 3.1, we have:

$$\begin{aligned} J^*(\theta_1, \theta_2) &= \mathbb{E}_{S_1, S_2} [T_1^*(\theta_1 | S_1) T_2^*(\theta_2 | S_2)] \\ &= \sum_{n_1, n_2 \geq 0} \pi_{n_1 n_2} \left(\frac{\mu_1}{\mu_1 + \theta_1} \right)^{n_1+1} \left(\frac{\mu_2}{\mu_2 + \theta_2} \right)^{n_2+1} \end{aligned}$$

The equilibrium probabilities π are the standard product-form solution [7, 5], which is most easily derived by RCAT. In fact, an added advantage of RCAT is that it constructs the required reversed process for node 2. Here, we already know what this process is – the same M/M/1 queue – but we do not know this for general nodes, even G-queues (with negative customers) [1, 4].

The above result generalises inductively to overtake-free paths in both open and closed networks to give the following:

Proposition 3.2 *For overtake-free path $\mathbf{z} = (z_1, z_2, \dots, z_m)$ in a queueing network of M nodes with state space \mathcal{S} at equilibrium ($1 \leq m \leq M$), the LST of*

the joint sojourn time probability distribution is

$$J^*(\theta_1, \dots, \theta_m) = \sum_{(n_1, \dots, n_M) \in \mathcal{S}} \pi_{n_1, \dots, n_M} \prod_{j=1}^m \left(\frac{\mu_{z_j}}{\theta_j + \mu_{z_j}} \right)^{n_{z_j} + 1}$$

where π_{n_1, \dots, n_M} is the equilibrium probability distribution of the network's state immediately prior to the instant of arrival of a task at any node.

Notice that π_{n_1, \dots, n_M} is well defined by the arrival theorem [5], being the same as an open network's steady state probabilities (at a random time point) or the steady state probabilities of a closed network with population reduced by one, depending on whether the network in question is open or closed, respectively.

In the case of open networks, π_{n_1, \dots, n_M} is a product of the form $\pi_1(n_1) \dots \pi_M(n_M)$ where $\pi_i(n_i) = (1 - x_i)x_i^{n_i}$ for some constants x_i , and so the result simplifies to

$$J^*(\theta_1, \dots, \theta_m) = \prod_{j=1}^m \frac{\mu_{z_j}(1 - x_{z_j})}{\theta_j + \mu_{z_j}(1 - x_{z_j})}$$

This is consistent with the fact that in a tandem series of stationary M/M/1 queues with fixed-rate servers and FCFS discipline, the sojourn times of a given task in each queue are independent. Interestingly, the proof of this result uses properties of reversibility and so we include it as an appendix, [8]. There is one obvious generalisation: the final queue in the series need not be M/M/1 since we are not concerned with its output. Also, the same result holds, by the same reasoning, when the final queue is M/G/c for $c > 1$. This contrasts with a similar result we get with our alternative approach in the next subsection.

In either approach, we observe that if service rates varied with queue length, we could not ignore tasks behind a given tagged task, even when they could not overtake, because they would influence the service rate received by the tagged task. Except in special cases, therefore, constant service rates are required.

3.2 G-networks

Suppose we have a tandem network comprising an M/M/1 queue and a G-queue that has an additional external arrival stream of negative tasks that remove the last task in the FCFS queue when it is non-empty [1]. If the G-queue is the first node, the conditions in section 2 are satisfied since the network is separable (by RCAT [4]), the second node is a reversible M/M/1 queue and the response time in the first queue depends only on the initial state. The response time distribution therefore has LST which is the product of that for the G-queue and that for the M/M/1 queue with arrival rate equal to the positive throughput from queue 1, i.e. the product of the external positive arrival rate and the probability of a task not being 'killed'.

Now suppose the G-queue is the second node, node 1 being M/M/1. The network is still separable and the sojourn time at the first node depends only on the initial state. However, the reversed (and forwards) sojourn time at the second node depends on the evolution of the first node since synchronised transitions with it influence the passage of the tagged task.³ The reversed node

³In the forwards process, arrivals from the first node offer protection from the negative arrivals at node 2; see [6].

2 process can be determined using RCAT and, given initial state (n_1, n_2) , the reversed response time is the time for $n_2 + k$ departures to take place, given $k \geq 0$ reversed negative arrivals (that increase the reversed queue length). This is a complex, transient calculation, analogous to that of [6]. The case with both nodes being G-queues is even more complex, the actual solution involving a Fredholm integral equation of the second kind.

It is interesting to note that the separable case has the G-queue as node 1 whereas an M/G/1 queue must be second when paired with an M/M/1. This was highlighted as unexpected in [6] but is routine to show in the new approach. Notice that if an M/G/1 queue were paired first with an M/M/1 queue, with FCFS queueing discipline, the network is not separable – it has long been known that no product-form then exists for the equilibrium queue length probabilities, and the conditions of RCAT correspondingly fail.

4 Generalised networks

The most obvious route to finding a new separable response time distribution in a network of two nodes requires that the network satisfy RCAT (so the reversed process is separable) and that the reversed sojourn time is tractable at the second node. This is not going to be an easy task since the first requirement would itself imply a new product-form. Nevertheless, suppose the first node is a queue with Erlang-2 service time (sum of two identical exponential random variables) and the second M/M/1. The reversed sojourn time at a reversed M/M/1 queue poses no problem, as above, and the sojourn time distribution in node 1 can be calculated as a mixture of Erlang distributions, using the initial steady state queue length probabilities. However, these probabilities are not simple, nor even known in closed form, and worse still, RCAT does not apply, so the network is unlikely to be separable.

We therefore seek to find a modified network which satisfies RCAT, for example creating additional external arrivals at node 2 by ensuring that all states in node 1 have an incoming, active, synchronising action type, using invisible actions [4]. If such a modification can be found and RCAT satisfied, the joint sojourn time distribution would indeed follow; the work proceeds.

5 Conclusion

Response times distributions – more generally, joint node-sojourn time distributions – can be derived much more simply and generally than previously using the reversed process of a separable network. In this way, most of the known separable solutions for the LSTs of response time distributions in queueing networks can be obtained. Moreover, many other special cases of product form solutions can be explained. Although it appears problematic at present to find completely new separable solutions, the methodology provides a handle for such problems and certainly is conducive to automation. In fact, new product-forms for equilibrium state probabilities could provide a basis, since they would at least, via RCAT, provide the right, separable reversed node processes.

References

- [1] E. Gelenbe. Random neural networks with positive and negative signals and product form solution. *Neural Computation*, 1(4):502–510, 1989.
- [2] P.G. Harrison. The distribution of cycle times in tree-like networks of queues. *Computer Journal*, 27(1):27–36, 1984.
- [3] P.G. Harrison. Turning back time in markovian process algebra. *Theoretical Computer Science*, 290(3):1947–1986, January 2003.
- [4] P.G. Harrison. Compositional reversed Markov processes, with applications to G-networks. *Performance Evaluation*, December 2004.
- [5] P.G. Harrison and Naresh M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1992.
- [6] P.G. Harrison and E. Pitel. Response time distributions in tandem G-networks. *Journal of Applied Probability*, 32:224–246, 1995.
- [7] J.R. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, 1963.
- [8] F.P. Kelly. *Reversibility and stochastic networks*. Wiley, 1979.
- [9] F.P. Kelly and P.K. Pollett. Sojourn times in closed queueing networks. *Advances in Applied Probability*, 15:638–656, 1983.

A Appendix: Proof of independence in tandem M/M/1 queues

Proposition A.1 *In a tandem series of stationary M/M/1 queues with fixed-rate servers and FCFS queueing discipline, the sojourn times in each queue of a tagged task are independent.*

Proof First we claim that the sojourn time of a tagged task, C say, in a stationary M/M/1 queue is independent of the departure process before the departure of C . This is a direct consequence of the reversibility of the M/M/1 queue.

To complete the proof, let A_i and T_i denote C 's time of arrival and sojourn time respectively at queue i in a series of m queues ($1 \leq i \leq m$). Certainly, by our claim, T_1 is independent of the arrival process at queue 2 before A_2 and so of the queue length faced by C on arrival at queue 2. Thus, T_2 is independent of T_1 . Now, we can ignore tasks that leave queue 1 after C since they cannot arrive at (nor influence the rate of) any queue in the series before C , again because all queues have single servers and FCFS discipline. Thus, T_1 is independent of the arrival process at queue i before A_i and so of T_i for $2 \leq i \leq m$. Similarly, T_j is independent of T_k for $2 \leq j < k \leq m$. ♠

Multipath Distance Vector Zone Routing Protocol for Asymmetric Mobile Ad-Hoc Networks MDVZRPA

Idris Skloul Ibrahim
isi3@macs.hw.ac.uk

A. Etorban
etorban @macs.hw.ac.uk

Peter J.B King
pjbk@macs.hw.ac.uk

School of Mathematical and Computer Sciences (MACS)
Heriot Watt University at Edinburgh UK

Abstract— Most of the ad hoc routing protocols assume that all wireless networks are symmetric (bidirectional links). In reality any practical network has some links may be unidirectional and hence the network is asymmetrical rather than symmetrical. The presence of such links can reduce the performance of the existing protocol and could lead to network clogging. In this paper we introduce a Multipath Distance Vector Zone Routing Protocol for Asymmetric mobile ad-hoc networks (MDVZRPA), which is a modification to MDVZRP. It is a hybrid routing protocol assumes that all routes in the routing table are active and usable, unless a broken link has been reported or discovered for reducing control traffic.

In addition to adopting MDVZRP technique, MDVZRPA is designed to deal with both bidirectional and unidirectional links by adding a new field called Symmetric-link in each route to distinguish between the two link types.

Keywords: MDVZRP, Asymmetrical Networks, Unidirectional and Bidirectional links.

I. INTRODUCTION

In recent years, mobile computing has enjoyed a tremendous rise in popularity. The continued minimization of mobile computing devices and the extraordinary rise of processing power available in mobile laptop computers combine to put more and better computer-based applications into the hands of a growing segment of the population. Mobile devices, such as laptop computers, Pocket PCs, cellular phones, etc., are now easily affordable, and are becoming more popular in everyday life [14]. At the same time, network connectivity options for mobile hosts have grown tremendously. The markets for wireless telephones and communication devices are experiencing rapid growth. Projections have been made that, in nowadays there are more than billion wireless devices in use. With the availability of mobile computing devices, users often have a natural tendency to share information between them, even though it is not planned in advance and there is no infra structure available for connection, for example, workers at rescue scenes

and employees in a meeting room, or conference or business. Therefore, the wireless mobile ad hoc networks become the practical and conventional solution in such like situations, without requiring each user to connect to the internet or to a wide-area network to communicate with each other because of cost and time. This type of network is easy and fast of deployment, where the nodes are communicate with each other through wireless medium without any fixed infrastructure. Mobile ad-hoc network was also being named as MANET [1] by IETF The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers.

A wireless ad hoc network as a decentralizing network offers an easy and fast connection between collection of autonomous nodes or terminals by forming a multi hop radio network. Since the nodes communicate over wireless links, they have to contend with the effects of radio communication, such as noise, fading, and interference. In addition, the links typically have less bandwidth than in a wired network. Each node in a wireless ad hoc network functions as both a host and a router, and the control of the network is distributed among the nodes [11] [12].

In general, MANET topology is dynamic, because of nodes departure and new nodes arrival during the connectivity time among the nodes, and asymmetrical, because the nodes communicate over wireless links which forms a different transmission range. Hence, there is a need for efficient routing protocols to offer optimum routes during the network establishing time to allow the network nodes to communicate over multi hop paths. Some of MNET features are characteristic of the type of packet radio networks that were studied extensively in the 1970s and 1980s. In general, a multi-hop routing protocol is needed in a mobile ad hoc network, because two hosts wishing to exchange packets may not be able to communicate directly with each other because they are out of radio range [14]. Figure (1) shows a simple ad hoc network of four mobile nodes using different wireless transmission range interfaces. Node A and D are not included within the wireless transmission range of node C. Only node A is included within the wireless transmission range of node D, and node D is not included within the transmission range of node B, as indicated by the circle around A, B and C. Nodes B, C and D are all included within the wireless transmission range of node A. If B and D want to communicate with each other by exchanging packets, they may ask node A to forward packets for them because node A is within the overlapped wireless transmission range between node B and node D.

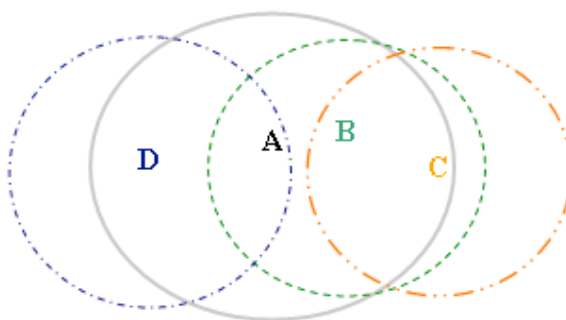


Fig (1): MANET with four wireless Mobile Nodes

In any practical MANET, packets are travel over one or more hops from one node to another node as demonstrated in Figure (1). In reality, the routing problem may be even more complex than this example, because of the nodes different wireless transmission range, and the network topology features which is dynamic because any or all of the nodes associated with the network may move at any time [14]. To provide routing the conventional way in MANET is to make each mobile node take the role as a router, and apply an existing routing protocol between them [16][17]. The fundamental difference between MANETs and the traditional wired networks that is the wired networks topology is stationary and static. This made the traditional protocols such as TCP/IP are not suitable for MANE and leads us for a specific requirement and constraints to provide routing protocols in such dynamic environments.

Over the few last years, many routing protocols have been proposed, where most of these protocols are based on distance vector or link state algorithms. In distance-vector routing protocols (i.e. DSDV) [4], routing information are periodically advertised to all nodes to get an up-to-date view over the entire network. Each node during the network establishing time sends to and receives from, all its neighbor nodes periodic messages and routing information to build and update its routing table, which contains the distance from itself to all possible destinations. Each node can decide whether to keep or update the next hop as the best and shortest path from itself to the specified destination based on comparison of the distances obtained from its neighbors. When each node has a packet to send to some destination, it simply forwards the packet to the decided next hop router. The advantage of this approach is that routes between arbitrary source - destination pairs are readily available, all the time, while the disadvantages are that the routing tables will occupy a large amount of space if the network is large, and that the updates may lead to inefficient usage of network resources if they occur too frequently.

Since ad-hoc networks are bandwidth limited and their topology changes often, an Optimized Link-State Protocol (OLSR) [5] has been proposed. While being suitable for small networks, some scalability problems can be seen on larger networks. The need to improve convergence and reduce control traffic has led to algorithms that combine features of distance-vector and link-state schemes. Such a protocol is the wireless routing protocol (WRP) [10], which eliminates the counting-to-infinity problem and avoids temporary loop without increasing the amount of control traffic. [11, 12]

In addition to the view point categorizing routing protocols in terms of either distance vector or link state routing, routing protocols for MANET also can be classified as uniform, non uniform or reactive routing protocols versus proactive routing protocols. In the reactive routing approach, a node initiates a route discovery (Route request) only when want to communicate with a destination which has no available route to it in its routing table, in other words, a routing protocol does not initiate route request until it is needed (Route On Demand). AODV [2], DSR [14], and TORA [13] are the most famous reactive routing protocols for MANET. The disadvantages of such algorithms are high latency time in route finding and excessive flooding can lead to network clogging (Blocking). On the contrary, the proactive routing approach is based on the exchange of knowledge of network topology periodically [9]. The proactive protocols provide a

needed route instantly at the expense of bandwidth because of transmitting periodic updates of topology frequently.

Hybrid routing protocols also exist and they try to achieve an efficient balance between both categories of protocols, where combining both the proactive and the reactive approach. ZRP is an example of hybrid routing protocols, was introduced in 1997 by Haas and Pearlman [6][7]. A more fine grained classification of ad-hoc routing protocols and taxonomy for comparing them can be found in [15].

I. SYMMETRIC AND ASYMMETRIC NETWORKS

In a symmetric computer network, all nodes can transmit and receive data at equal rates. Asymmetric networks, on the other hand, support disproportionately more bandwidth in one direction than the other. This can be a problem in wireless networks which adopt a TCP technique where TCP relies on ACKs for reliable delivery and for congestion control. If ACKs are not reliably returned the smooth of packets will be disrupted by retransmissions. Most of ad hoc networks protocols have been designed assuming that the underlying technology was bidirectional (*Symmetrical Network*). As an example, a set of nodes which are connected through a single physical network assume they can exchange routing information with each other as shown in figure (2). Exchanging routing information enables the discovery of the underlying network topology, and the routing traffic via discovered networks.

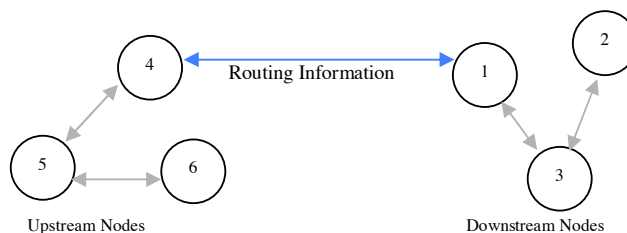


Fig (2): A Symmetric Network

However, if the link connecting these nodes is unidirectional (*Asymmetrical Network*), we can say that all downstream nodes have received only capabilities and therefore cannot send routing information to upstream nodes as shown in figure (3). As a result, upstream nodes cannot discover downstream network topologies dynamically and will therefore never forward information towards them.

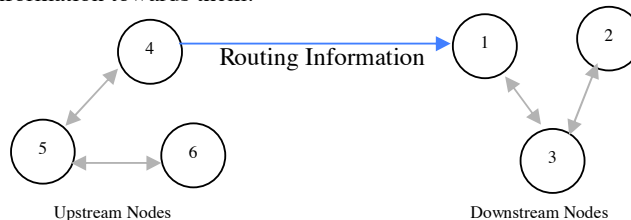


Fig (3): An Asymmetric Network

Generally, in the presence of a unidirectional link, many routing protocols, will fail to operate and lose to send data therefore, to provide full network connectivity we need to make the node to discover if the link is a bidirectional or unidirectional link before sending over any data.

II. MDVZRPA: MOTIVATION

A node in MDVZRPA has a flat view over the entire network when it joins the network and broadcasts a beacon message for the first time. It is easy to get ready multipath to each destination in the entire network by unicasting and receiving routing information (*full dump*) from all its bidirectional neighbours to build its own routing table. Since all nodes proactively store local routing information, route requests can be more efficiently performed without querying all the network nodes. In case of receiving an error message regarding to a broken active link or non reachable node, MDVZRPA uses an alternative path getting technique to get a suitable alternative path (*Best metric*) among the multipath that were stored into the node routing table, instead of wasting time in route repair or route request every time. Also MDVZRPA uses a field called *Symmetric* to distinguish between unidirectional and bidirectional links. Once a node receives the *Hello* message from a new node, it adds an entry in its routing table to this destination (new node) assuming the link between them as an unidirectional link by resetting the *Symmetric* field=0. The *Symmetric* field is set to 1 when a routing information is received from the new node, then the link is considered as a bidirectional link.

III. MDVZRPA: ZONE RADIUS IN AN ASYMMETRICAL NETWORK

The zone radius is the distance in number of hops from the specific node to the last node in its zone. A routing zone is defined for each node separately, and the zones of neighboring nodes overlap. The routing zone has a radius R expressed in hops. The zone thus includes the nodes whose distance from the node in equation is at most R hops. Figure (4) shows a new node (i.e. node 7), and its routing zone when it joined the network. Each node has only one hop from the new node we call it a 1st hop neighbour where radius $R=1$ (i.e. 4, 5 and 6), while any node has 2 hops from the new node, we call it 2nd hop neighbour where $R=2$ and so on. If a node has number of hops (*Distance*) from source node = zone radius, then we call it a peripheral node (i.e. 2, 3). All the rest nodes which have distance $> R$ (i.e. 1) are called *out of zone* nodes.

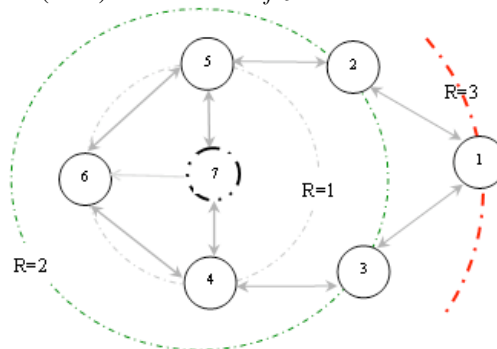


Fig (4): Zone Radius, where $R=1, 2$ or 3 in an Asymmetrical Network

IV. MDVZRPA: ROUTING INITIALIZATION

During the initialization stage as each node joins the network, it adds an entry to itself in its routing table, and broadcasts a periodic beacon (*Hello message*). In figure (5) we assume node 6 is a new node joined an asymmetrical network, where the node zone radius is 2.

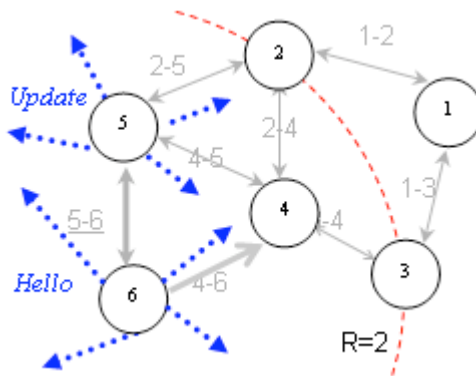


Fig (5): A Hello Message along an Asymmetrical Network

A node which receives the *Hello* message (*i.e.* 4, 5) checks if it has a direct route (*hop=1*) to the *Hello* message sender. If so, it updates the entry regarding to the next *Hello* message expecting time (*time-out field*), and discards the *Hello* message (*the neighbor is still there*). Otherwise, it adds a new route entry where the route *destination* and 1st *hop* fields are the address of the node that sent the *Hello* message as shown in Table (1), while the *link-id* field is the *Hello* message receiver - *Hello* message sender addresses (4-6, 5-6), sets number of hops field (*Metric*) to 1, sets *time-out* field, and resets the *Symmetric* field assuming the link between them as an unidirectional link (unusable) at the beginning. Then, it unicasts its routing information to the new node (*Full dump*), if a full dump back message is received from the new node, the node sets the *Symmetric* field considering the link between them as a bidirectional (usable) and broadcasts an *Update* route message to its neighbors regarding to the new node. Table (1) shows new routes to the new node (6) added by node 4 and 5 in their routing table respectively.

Node 4:

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric
6	6	-	1	4-6	0

Node 5:

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric
6	6	-	1	5-6	0

Table (1): Routes are obtained after receiving a *Hello* message

In case of node 4, the link 4-6 is unidirectional as shown in figure (5). Therefore, node 6 will not receive routing information (*full dump*) message from node 4, it is still unknown for node 6. Hence, node 6 will not send back its routing information to node 4.

In case of node 5, the link 5-6 is bidirectional as shown in figure (5). Therefore, node 6 will receive the routing information (*full dump*) from node 5, and unicasts its routing information (*full dump*) to node 5. Node 5 sets the *Symmetric* field, considering the link between them as a bidirectional and broadcasts an *Update* message regarding to the new node to its 1ST hop neighbors (node 2, 4) as shown in figure (5), where these nodes are the 2nd hop neighbors of the new node using node 5 as a 1st hop.

The new node 2nd hop neighbours add an entry in their routing tables and discard the *Update* message, where the *destination* and 2nd hop fields of the entry are the address of the node that sent the *Hello* message (node 6), the 1st hop field is the address of the node that sent the *Update* message (node 5), while the *link-id* is the same as the link id included in the *Update* message 5-6, and the metric is incremented by 1 as shown in table (2). The *Update* message is discarded and not propagated when the metric equals the zone radius (**R**).

Node 2:

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric
6	5	6	2	5-6	1

Node 4:

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric
6	6	-	1	4-6	0
6	5	6	2	5-6	1

Table (2): Routes are obtained from the *Update* message sent by node 5

Further more, Node 6 continues to broadcast a periodical beacon (*Hello*) from time to time. Each time node 5 receives this beacon, it finds a direct route (*hop=1*) in its routing table to node 6 where the link is a bidirectional (usable), therefore, it only updates the entry belongs to node 6 regarding to the next *Hello* message expecting time (*time-out field*) and discards the message. Also, node 4 finds a direct route (*hop=1*) to node 6 where the link is unidirectional (unusable), therefore, it checks if it has indirect route to node 6. If indirect route is found, such as in this case where node 4 has a route to node 6 through node 5 as shown in table (2), it updates the entry belongs to node 6 regarding to time-out field and unicasts a routing information message to node 6 using that route. Otherwise, it only updates the time-out field and discards the message.

Once node 6 received the routing information of node 4, it adds a direct route where *Symmetric* field =1(useable), without paying attention to the next *Hello* message expecting time (*time-out field*) of node 4. In other words, node 6 doesn't consider node 4 as a neighbor.

The new node immediately starts to build its routing table entry by entry, excluding any similar, long and joint paths, once received a routing information message. Tables 3-8 are the routing tables of the entire network nodes after the new node joined the network.

Destination	1 st hop	2 nd hop	Metric	link-id	Symmetric	Notes
1	1	-	0	1-1	1	Initialization
2	2	-	1	1-2	1	...
3	3	-	1	1-3	1	...
4	2	4	2	2-4	1	Multipath
4	3	4	2	3-4	1	Multipath
5	2	4	3	4-5	1	Update

Table (3): Routing table of node 1

Destination	1 st hop	2 nd hop	Metric	link-id	Symmetric	Notes
1	1	-	1	1-2	1	...
2	2	-	0	2-2	1	Initialization
3	1	3	2	1-3	1	...
3	4	3	2	3-4	1	...
4	4	-	1	2-4	1	...
4	5	4	2	4-5	1	...
5	4	5	2	4-5		...
6	5	6	2	5-6		Update

Table (4): Routing table of node 2 after receiving the *Update* messages

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	1	-	1	1-3	1	...
1	2	2	2	1-2	1	...
2	4	2	2	2-4	1	...
3	3	-	0	3-3	1	Initialization
4	4	-	1	3-4	1	...
5	1	2	3	2-5	1	...
5	4	5	2	4-5	1	...

Table (5): Routing table of node 3 after receiving the *Update* messages

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	2	1	2	1-2	1
1	3	1	2	1-3	1
2	2	-	1	2-4	1
3	3	-	1	3-4	1
4	4	-	0	4-4	1	Initialization
5	5	-	1	4-5	1
6	5	6	2	5-6	1	Update
6	6	6	1	4-6	0	Hello

Table (6): Routing table of node 4 after receiving the *Update* messages

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	2	1	2	1-2	1	...
2	2	-	1	2-5	1
3	4	3	2	3-4	1
4	4	-	1	4-5	1
4	6	4	2	4-6	1	Update 6
5	5	-	0	5-5	1	Initialization
6	6	-	1	5-6	1	Hello

Table (7): Routing table of node 5 after receiving the *Update* messages

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	5	2	3	1-2	1	Full dump 5
2	4	2	2	2-4	1	Full dump 4
2	5	2	2	2-5	1	Full dump 5
3	4	3	2	3-4	1	Full dump 4
4	4	-	1	4-6	1	Full dump 4
4	5	4	2	4-5	1	Full dump 5
5	5	-	1	5-6	1	Hello
5	4	5	2	4-5	1	Full dump 4
6	6	-	1	6-6	1	Initialization

Table (8): Routing table of the new node (6) after receiving the full dump

From the previous tables we can see that each node has one or more entries for each destination in the network, except node 1 and 3. They have no routes to node 6, because both are outside of node 6 routing zone. Therefore, both should initiate a route request *on demand* when need to send data to node 6.

V. MDVZRPA: ROUTE ON DEMAND

If a node needs to communicate with another node in the network and it has no route available in its routing table to that node because it is outside its routing zone, in this example we assume that node 3 as a *Source* node needs to communicate with node 6 as a *Destination*. The *S* node broadcasts a route request message *RREQ* with the *D* address to find a route to the required destination, as shown in figure (6). A route can be determined when the route request *RREQ* reaches a node that offers accessibility to the destination, (e.g., one of the destination's 1st hop neighbours node 1, 4 or one of the peripherals nodes 2, 5). As shown in figure (6) and from table (6), node 4 has two routes in its routing table to the Destination. The shortest route is in 1 hop distance from node 6, where *link-id* is 4-6, but it is asymmetric link (*Symmetric*=0), therefore it is useless. The second route is longer, in 2 hops distance from node 6, where *link-id* is 5-6. It is better because the link is symmetric (*Symmetric*=1). Also from table (4) node 2 has a route in 2 hops distance to the destination. The route is made available by unicasting a *RREP* back to the source node and is written in its routing table. The source node has got two routes to the required

destination the first route in 3 hops distance through node 4 as a 1st hop, node 5 as a 2nd hop where the *link-id* is 5-6. The second route in 4 hops distance through node 1 as a 1st hop, node 2 as a 2nd hops and the *link-id* is 5-6. But because the two routes are node joint (*node 5*) and link joint (5-6), then node 3 chooses only one of them, the shortest one as shown in table (9).

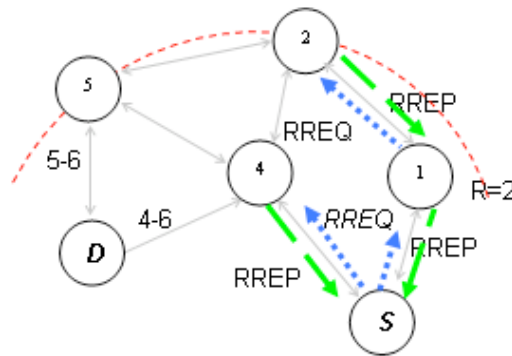


Fig (6): Route on Demand along an Asymmetrical Network

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	1	-	1	1-3	1	...
2	1	2	2	1-2	1	...
2	4	2	2	2-4	1	...
3	3	-	0	3-3	1	Initialization
4	4	-	1	3-4	1	...
5	1	2	3	2-5	1	...
5	4	5	2	4-5	1	...
6	4	5	3	5-6	1	RREP

Table (9): Routing table of node 3 after RREP message

VI. MDVZRPA: BROKEN LINKS DISCOVERY

A node discovers the broken link between itself and its neighbour using the *time-out* field. It is the time in which the node expects to receive a Hello message from that neighbor confirming that it is still exists (reachable). This field is set to the expected next hello message time once the node received the *Hello* message from that neighbor. If the node didn't receive a *Hello* message in the expected time then, the node considers that neighbor unreachable. In this case the node assigns any entry in its routing table where the *destination* or 1st *hop* fields are equal to the address of that neighbor as a broken link, by setting the *Metric* field to infinity as shown in tables (10, 11), and then generates and broadcasts a route error message (*RERR*) carries the *link-id* of the broken link. Each node receives this message, checks if it has any entry with the same *link-id* to assign it as a

broken link as shown in tables (12, 13). The next section is an example for node movement and broken link in details.

VII. MDVZRPA: NODE MOVEMENT

Node movement is one of the biggest challenges in MANET, where mobile nodes cause broken links as they move from place to place. Any node that discovers a broken link should generate and broadcast a forwarded route error *RERR*, where the nodes update their routing tables regarding to that error message. In figure (7), we assume that node 6 has moved away, both node 5 and 6 discovered that the link-id between them is broken, each node (5, 6) searches for the direct route to the other in its routing table to get its *link-id* and assigns it as a broken link, assigns any route it has where the 1st hop field is the non reachable node as shown in tables (10, 11) and then, broadcasts *RERR* message carrying the non reachable node address with the link-id (5-6) to be assigned by any neighbor it has a route carrying the same link-id in its routing table.

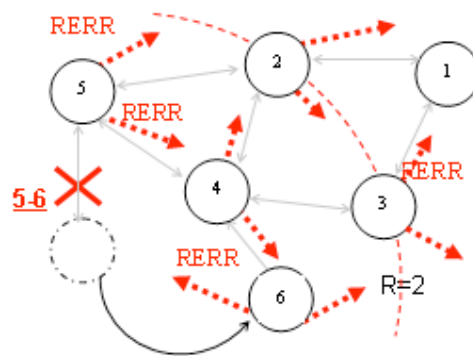


Fig (7) Route Error Message along an Asymmetrical Network

Each node receives *RERR* message, assigns any entry in its routing table with the same link-id as shown in tables(12, 13), and rebroadcasting the same error message *RERR* and so on, unless the node has no route carrying that link-id, in this case it discards the *RERR* message.

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	2	1	2	1-2	1	...
2	2	-	1	2-5	1	...
3	4	3	2	3-4	1	...
4	4	-	1	4-5	1	...
4	6	4	∞	4-6	1	Deleted
5	5	-	0	5-5	1	Initialization
6	6	-	∞	5-6	1	Deleted

Table (10): Routing table of node 5 after discovering the broken link

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	5	2	∞	1-2	1	Deleted
2	4	2	2	2-4	1	Full dump 4
2	5	2	∞	2-5	1	Deleted
3	4	3	2	3-4	1	Full dump 4
4	4	-	1	4-6	1	Full dump 4
4	5	4	∞	4-5	1	Deleted
5	5	-	∞	5-6	1	Deleted
5	4	5	2	4-5	1	Full dump 4
6	6	-	1	6-6	1	Initialization

Table (11): Routing table of node 6 after discovering the broken link

Destination	1 st hop	2 nd hop	Metric	link-id	Symmetric	Notes
1	1	-	1	1-2	1	...
2	2	-	0	2-2	1	Initialization
3	1	3	2	1-3	1	...
3	4	3	2	3-4	1	...
4	4	-	1	2-4	1	...
4	5	4	2	4-5	1	...
5	4	5	2	4-5		...
6	5	6	2	5-6		Deleted

 Table (12): Routing table of node 2 after receiving the *Update* messages

Destination	1 st hop	2 nd hop	Metric	Link-id	Symmetric	Notes
1	2	1	2	1-2	1
1	3	1	2	1-3	1
2	2	-	1	2-4	1
3	3	-	1	3-4	1
4	4	-	0	4-4	1	Initialization
5	5	-	1	4-5	1
6	5	6	∞	5-6	1	Deleted
6	6	6	1	4-6	0	Hello

 Table (13): Routing table of node 4 after receiving the *RERR* message

VIII. CONCLUSIONS AND FUTURE WORK:

In this paper, we proposed MDVZRPA a multipath routing protocol for asymmetric mobile ad-hoc networks. It is a development to our previous protocol MDVZRP [8] for symmetric mobile ad-hoc networks, which is extend to our MDSDV [3] protocol. MDVZRPA is a proactive for all destinations inside the routing zone and reactive for destinations outside the routing zone, supports both bidirectional and unidirectional links. It uses broadcast and unicast techniques to send the packets over both links. Nodes create their routing tables to save multi optimum paths using *Hello* and *Full dump* messages, where maximum number of optimum routes depends on number of neighbors of the

source node. Nodes update their routing tables using Update route, Route request (*RREQ*), Route replay (*RREP*) and Route Error (*ERR*).

If a node wants to initiate communication with a node to which it has no route, MDVZRPA will try to establish such a route using route request mechanism. The protocol allows sending packets by alternative paths in case of the primary path breaks; when a node discovers a broken link to one of its neighbors, broadcasts an error message using the *link-id* to identify the unreachable node. Any node receives the broken link error assigns the right route to that node using the *link-id* included in the error message as unusable route (∞). MDVZRPA gives the node ability to get information from any route pass through it.

In our future work, we are going to evaluate and compare MDVZRPA to DSDV, AODV and ZRP according to the following evaluation metrics (data throughput, packet delivery ratio, routing overhead and average packet delay).

REFERENCES:

- [1] Joseph E Macker, M. Scott.: Mobile Ad Hd Networking and the IETF, Information Technology Division, Naval Research Laboratory Washington, DC, USA Institute for Systems Research, University of Maryland, College Park, MD, USA
- [2] Perkins, C. E., Royer, E. M.: Ad-hoc On-Demand Distance Vector Routing, February 1999, Proc. 2nd IEEE Workshop on Mobile Computer Systems and Applications, pp. 90-100
- [3] Peter J.B King, A. Etorban and Idris Skloul Ibrahim: Multipath Destination Sequenced Distance Vector (MDSDV), 8th PG Net, Liverpool John Moores University, UK 28-29 June 2007, pp. 93-98.
- [4] Perkins, C. E., Bhagwat, P.: Highly Dynamic Destination-sequenced Distance-Vector Routing (DSDV) for Mobile Computers, October 1994, Computer Communications, pp. 234-244
- [5] Clausen, T., Jacquet, P., Laouiti, A., Minet, P., Muhlethaler, P., Qayyum, A., Viennot, L.: "Optimized Link State Routing Protocol", September 2001, IETF Internet Draft, draft-ietf-manet-olsr-06.txt
- [6] Nicklas Beijar.: Zone Routing Protocol (ZRP), Networking laboratory, Helsinki University of Technology P.O. Box 3000, FIN-02015 HUT, Finland, Nicklas.Beijar@hut.fi
- [7] Haas, Zygmunt J., Pearlman, Marc R.: A New Routing Protocol for The Reconfigurable Wireless Networks, Proceedings of 6th IEEE International Conference on Universal Personal Communications, IEEEICUPC'97, San Diego, California, USA October 12-16, 1997
- [8] Idris Skloul Ibrahim, A. Etorban and Dr Peter J.B King: Multipath Distance Vector Zone Routing Protocol (MDVZRP), 9th PG Net, Liverpool John Moores University, UK 23-24 June 2008.
- [9] Jacquet and Paul Muhlethaler and Thomas Clausen and Anis Laouiti and Amir Qayyum and Laurent Viennot: Optimized Link State Routing Protocol, IEEE INMIC 01, 28-30 December 2001, Lahore, Pakistan, pp. 62-68,
- [10] Murthy, S., Garcia-Luna-Aceves, J. J.: An Efficient Routing Protocol for Wireless Networks, October 1996, MONET, Vol 1, No 2, pp. 183-197
- [11] Haas, Zygmunt J., Pearlman, Marc R.: Providing Ad-hoc connectivity With Reconfigurable Wireless Networks, Ithaca, New York, May 1998.
- [12] Pearlman, Marc R., Haas, Zygmunt J.: Determining the Optimal Configuration for the Zone Routing Protocol, August 1999, IEEE Journal on Selected Areas in Communications, Vol. 17, No. 8, pp. 1395-1414
- [13] Park, V. D., Corson, M. S.: A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, April 1997, Proc. IEEE INFOCOM '97, Kobe, Japan
- [14] Johnson, D. B., Maltz, D. A.: Dynamic Source Routing in Ad-Hoc wireless Networking, Mobile Computing, T. Imielinski and H. Korth, Eds. Norwell, MA: Kluwer, 1996, pp. 153-181

- [15] L. M. Feeney. A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks, October 1999, SICS Technical Report
- [16] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, London, UK, Aug. 1994, pp. 234–244.
- [17] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," Proc. IEEE, vol. 75,no. 1, pp. 21–32, Jan. 1987.

Guided Subsystem Performance Assessment Architecture for Grid Service

Jin Wu, Yichao Yang, Yanbo Zhou, Linghang Fan and Zhili Sun*

Abstract

In a typical Grid scenario, multiple component services are autonomously federated into a composite Grid service based on the workflow expressed by process description models. For the simplicity to resource sharing, Grid services only expose a standard interface to users. The encapsulated appearance of Grid service hides almost all system details from users. However for some applications, when users want to know more about the performance of the service in planning phase, problem emerges. They simply have not means in knowing how well the system will cope with their demands. Therefore, we say that once the workflow is expressed by the user, performance assessment service should be in place to estimate the likeness of under-performing of the service and let users be notified before their subscription decision. Since a Grid service always involve multiple service provides and its failure may comes from different sources, Grid performance assessment aims to consider all involved factors and give an overall likeness of under-performing estimation. We believe current Grid infrastructures do not integrate adequate performance assessment measures to provide such function. Therefore, we present in this paper the architecture of Guided Subsystem Approach for Grid performance assessments. Based on this architecture, the performance assessment measures can be given by the Grid platform to extend its functionality.

1. Introduction

More and more applications are applied on the Grid infrastructure. However, the lack of performance indications becomes an obvious obstacle for the continuously promotion of the Grid. User's sceptics in service quality significantly hold back the efforts of putting more applications onto Grid: it is hard to convince users to relay on the Grid infrastructure before they have been clearly notified with the service quality they will receive, especially when all Grid services are encapsulated with only a standard interface be exposed. Such a problem was caused by the nature of Grid. The Grid middleware hides system details by allowing users to access service through a portal without recognising the details of resource providers. On one hand, a standardised resource usage interface provides an easy access to remote resources. On the other hand, the system details are hidden behind for user to aware the performance of Grid services. This paper studies the performance assessment architecture to relief

* Centre for Communication System Research (CCSR), University of Surrey, UK, {jin.wu; y.yang; y.zhou; l.fan; z.sun}@surrey.ac.uk

the problem. For measuring perception, performance assessment process is introduced in the planning phase of Grid applications. The performance assessment is a user-centric effort. It collects the requirement from the user, and feedbacks how well the system will cope with the user's requirement after examine related Grid components. This component does not try to improve the performance of individual component, but assist the users to select to services more wisely by enabling the comparison among available services providers on like-for-like bases.

Theoretically, two approaches can be used to obtain the performance assessment of an application: the user initiated assessment, and the infrastructure based assessment. The user initiated assessment approach evaluates the performance of an application by users' own efforts. The services from all providers are trialled one by one, and their performances are documented for like-for-like comparisons. While this approach is easy to use, its utility is generally limited by its overhead, accuracy, and management concerns. For instance, a large number of users making independent and frequent assessment trials could have a severe impact on the performance of services themselves. Furthermore, the assessment abilities at user-ends are always limited which weakens the accuracy of assessment results. More important still, application providers might not agree to open their systems for trials for a serial of economical and security concerns. Ideally, dedicated modules attached to service providers should be in place while its assessment results could be made available, with low overhead, to all perspective users. This is the basic motivation of infrastructure based performance assessment for Grid services. Figure 1 illustrates the relationships behind the infrastructure based performance assessment approach in Grid environment. User subscribes to services in Grid environment instead of directly calling physical components.

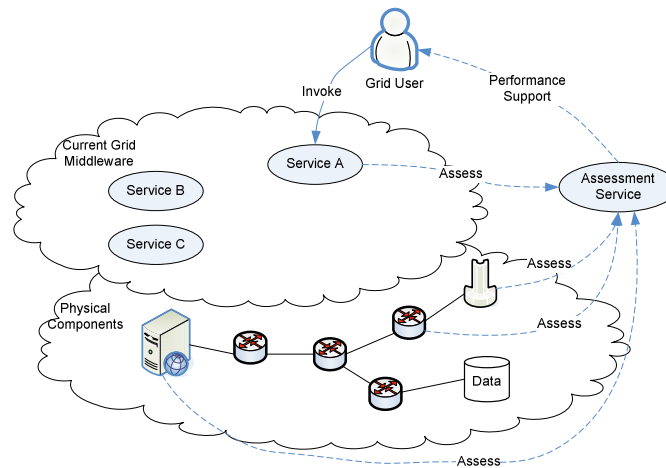


Figure 1. Infrastructure Based Performance Assessment

A performance assessment service is in place to evaluate the performance of general Grid services. It relies on an infrastructure to obtain the running status of physical components which could support the Grid services. When a user wants to know the possible performance of a given service, the performance assessment service is able to give the support to expected performance based on the physical component

performance assessment and workflow of the service. By applying the infrastructure based performance assessment approach, the following advantages can be gained:

- Physical components are opened for assess to a certain number of authorised assessment programs;
- Performance assessment results are shared among multiple users;
- Users receive swift reply of service performance estimation.

This approach settles the efficiency and security concerns.

Current researches have limited emphasises on this topic. Previous researches have discussed the use of infrastructure to obtain data transmission performance between Internet hosts. A simple protocol for such a service, SONAR, was discussed in the IETF as early as February 1996 [2], and in April 1997 as a more general service called HOPS (Host Proximity Service) [3]. Both of these efforts proposed lightweight client-server query and reply protocols similar to the DNS query/reply protocol. Ref [4] proposed a global architecture, called "IDMaps", for Internet host distance estimation and distribution. This work propose an efficient E2E probing scheme that satisfies the requirements for supporting large number of client-server pairs. However, the timeliness of such information is anticipated to be on the order of a day and therefore, not reflect transient properties. Furthermore, this work can only assess the transmission delay among Internet host, this could be an underneath support of performance assessment, but this work did not touch the performance issue related to users. The topic of server selection has also been touched. Ref [5] proposed passive server selection schemes that collects response times from previous transactions and use this data to direct client to servers. Ref [6, 7] proposed active server selection schemes. Under these schemes, measurement to periodically probe network paths is distributed through the network paths are distributed throughout the network. Based on the Round Trip Time that is probed, an estimated metric is assigned to the path between node pair. Ref [8] study a scenario where multimedia Content Delivery Network are distributing thousands of servers throughout the Internet. In such scenario, the number of tracers and the probe frequency must be limited to the minimum required to accurately report network distance on some time scale. Our paper extends the research by combing the above two idea together - an assessment infrastructure is in place to assist the application selection. The novelties of this research lie on: a) the assessment infrastructure has been extended to support performance assessment other than network performance metrics assessment; b) a general architecture considering both application layer and transmission network performances are proposed; and c) Internet application users will be notified with possible performance before their executions. The reminder of the paper are organised as follows. Section 2 gives an architecture level study of the performance assessment. Section 3 studies the detailed procedures for the performance assessment identified in Section 2. Section 4 presents a case-study where the analysis in Section 3 is put into practice. Section 5 gives the discussions and future works.

2. Architectural Level Study to Performance Assessment

An architectural level study of performance assessment is given in this section. Performance assessment is used to find out the degree of likeliness by which the performance of a Grid service fulfils the user's requirements. It is expected that through performance assessment, users are facilitated with the ability to aware the quality of the Grid service before subscriptions. As discussed above, the Grid service

suffers uncertainties in terms of service qualities due to its distributed and service-oriented nature. In many cases, multiple component services from different administration domains are federated into a composite service which is finally exposed to users. Needless to say, the performances of related component services affect the performance of the composite service that is directly visible to users.

The accuracy, efficiency, and scalability are main concerns for the performance assessment of Grid services. Generally, Grid could be large scale heterogeneous systems with frequent simultaneous user accesses. Multiple performance assessment requests must be employed to assess performance in a timely and overhead efficiency manner. Apparently, the larger and more complex the system, and the more varied its characteristics, the more difficult it becomes to conduct an effective performance assessment. Subsystem-level approach is a solution to overcome the scalability problem to performance assessment over large scale systems. As its name suggests, the Subsystem-level approach accomplishes a performance assessment by developing a collection of assessments subsystem by subsystem. The rationale for such an approach is that the performance meltdown that user experienced will show up as a risk within one or more of the subsystems. Therefore, doing a good job at the subsystem level will cover all the important area for the whole system.

The following observations identify the limitations of the approach.

- Independent subsystem performance assessment tends to give static performance and assume in the way that the upcoming invoking request does not carry out a meaningful impact to the subsystem performance. But a subsystem can exhibit differently when the additional invoking request is applied. Without a dynamic performance assessment which takes the characteristics of both invoking request and subsystem, the performance assessment result has a systematic inaccuracy.
- Independent subsystem performance assessment is prone to variance in the way performance metrics are characterised and described. Without a common basis applied across the system, subsequent aggregation of subsystem results is meaningless or impossible to accomplish. A consequence is the inevitable misunderstanding and the reduced accuracy of performance assessment.
- Subsystem A's analysts can only depict the operation status and risk of Subsystem A, but it may not have an accurate understanding of Subsystem A's criticality to the overall service performance. As viewed from the perspective of the top level performance assessment, the result from subsystem can be wasted effort assessing unrelated performance metrics, or subsystem performance metric that crucial to the overall performance assessment is not measured.

Given the weakness of the conventional subsystem performance assessment approaches, we propose the Guided Subsystem Approach to performance assessment as an enhancement to conventional approaches.

The main idea of the Guided Subsystem Approach is to introduce a user-centric component that interprets the user's assessment requests and assessment function of subsystems. It conducts macro-level pre-analyse efforts. The top-down efforts are enforced to translate those performance assessment requests and transmit to the subsystem assessment modules. Such the dynamically generating of assessment requests to subsystem helps to prevent mismatching between user's assessment

requirements and the assessment actions applied onto the subsystems. The Guided Subsystem Approach takes the advantage of both top-down and subsystem approaches while presenting better efficiencies, consistencies, and systematic accuracies. Furthermore, the Guided Subsystem Approach has top-down characteristics together with characteristics from subsystem. The user centric interpretation components and subsystem performance assessment components can be designed and implemented separately.

Figure 2 gives an architectural level flow diagram for Guided Subsystem Approach generally applicable to the performance assessment process of most Grid services. Macro processes and sub-system processes are applied respectively to user-centric and system-centric assessments.

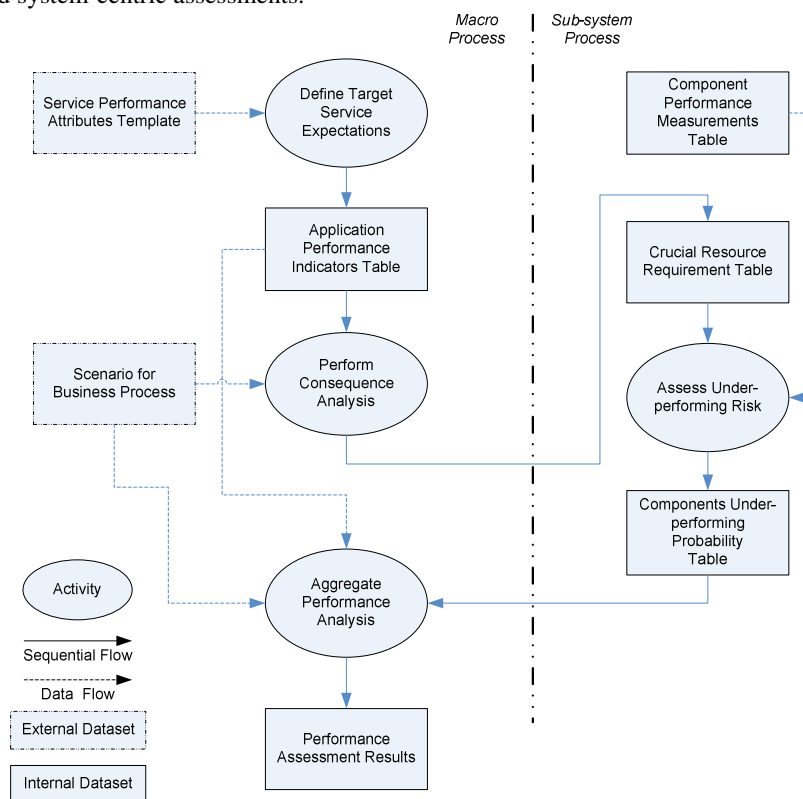


Figure 2. Architectural Level Process for Guided Subsystem Assessment

First, the user-centric assessment is used to identify and document a description of the user's expectations to include performance indicators of the target service. The Application Performance Indicators Table is compounds of performance attributes that can be used to depict, in a standard way, the performance a user expects from the target service. This activity is named as Define Target Service Expectations. As part of this activity, the user-centric performance assessment module gets the Service Performance Attributes Template and completes the attributes' definitions according to the user's performance expectations to the target service. The Service Performance Attributes Template is authored by system analysis of applications, and could be, nevertheless, updated thanks to users' feedbacks. However, the management and

standardisation of the Service Performance Attributes Template is outside the scope of this research. Without loss of generality, we simply consider standard Service Performance Attributes Template can be given by a third party generalised service when a target service is identified. The next step is to generate the Crucial Resource Requirement Table which depicts the performance requirement of each subsystem participate the business process of the target service. This activity indicates local constraints of resources based on the analysis of the global constraints given by the previous activity and the Scenario for Business Process. A Scenario for Business Process is a set of interactions among components triggered by specific input stimulate to make the business process of a service possible. Then, the crucial resource requirements are transferred to sub-systems for analysis.

When a Crucial Resource Requirement Table authored by the Marco process is received by the system-centric assessment modules, sub-system assessment process is activated. The Assess Under-performing Risk activity is used to measure how well the sub-system, usually another encapsulated component service, meets the resource requirement from the composite Grid service. The resource requirement can be read from the Crucial Resource Requirement Table. Also, it is assumed that the resource of the component can be obtained by a local agent who forwards the component's operation status to the Component Performance Measurements Table. The assessment result in the form of under-performing probability is forwarded to the Component Under-performing Probability Table and transmitted back to user-centric assessment module. The activity Aggregate Quality Analysis collects all performance measurements from sub-systems and gives performance assessment result.

It is expected that the Guided Subsystem Approach for Grid service performance assessment can improve the performance assessment in terms of scalability, accuracy and consistency.

3. Guided Subsystem Performance Assessment

We discuss in this section how to measure the performance of a given Grid service by applying the Guided Subsystem Approach. Basically, when user subscribes to a Grid service, it expects the Grid service to be delivered with some quality measures. Once the performance of the Grid service is worse than the minimum acceptable quality, then the Grid service is viewed as under-performing. As previously discussed, a Grid service may invoke multiple functional components spread across multiple administration domains. Performance malfunctioning of a Grid application takes place when related functional components supporting the application do not meet the user's requirements. We show in this section how the assessment be done following the process given in the last section.

3.1 Definition of Target Service Expectation

For an assessment infrastructure, it needs to give performance indications after users express their expectations to the target Grid service. Users can have diverse performance expectation for a Grid service. Their performance expectations should be expressed in a standardised way to let the assessment infrastructure learn the need of users. A performance indicator is defined to notify users the result of performance assessment. The performance indicator should have pragmatic meaning and allow

user to perform like-with-like comparison between similar Grid services delivered from different providers.

The concept of Expected Cost of Under-performing (ECU) is introduced in measuring the potential cost a user might require to pay for the lost due to performance dissatisfactions. ECU is a predicted value given by the performance assessment process. An accurate prediction of ECU can be used as an indication of performance. ECU is tightly coupled with performance: a lower ECU will result as better performance, and a higher performance will bring more risks. The value of ECU passes back to users to assist users in two aspects during the planning phase of remote application:

- ECU can be used as an indication to compare Grid services from different providers on like-with-like bases. One application with lower ECU can lead to a better performance than others;
- ECU measurement can help users to aware the potential risk of performance dissatisfaction for target Grid service.

For the purpose of measuring the ECU of a target Grid service, three types of inputs need to be identified:

- **Set of Performance Malfunctioning (SPM)** is a set containing any possible performance malfunctioning feature by which Internet application users might possibly be experienced. It is a framework of discernment attached to applications. The descriptions of this input identify the differences between applications.
- **Consequence of Performance Malfunctioning (CPM)** measures the damages every particular performance malfunctioning feature could possibly cause. It depicts the users' expectations of performance towards the application.
- **Probability of Performance Malfunctioning (PPM)** measures the possibility of performance malfunctioning appears under a given framework of discernment. This input depicts the how the underneath system reacts to the invoking requests.

The ECU of a target Grid service can be given upon the collection of the above three aspects of inputs.

To obtain the *SPM* is the first step for *ECU* measurement. The performance malfunctioning is a feature of an application that precludes it from performing according to performance specifications, a performance malfunctioning occurs if the actual performances of the system are under the specified values. *SPM* is usually defined when a type of Grid service is composed. When a type of Grid service is identified, its standard *SPM* can be obtained. Different types of Grid services could lead to different *SPMs*. Basically, *SPM* is authored by system analysis, and could be, nevertheless, updated thanks to users' feedbacks. However, the management and standardisation of *SPM* is outside the scope of this research. Without loss of generality, we simply consider standard *SPMs* can be given by a third party generalised service when a use-case is presented, where the *SPM* of service *u* is denoted as SPM_u . For a service *u* with a standard *SPM* including *n* malfunctioning type, there exists $SPM_u = \{s_1, \dots, s_n\}$, which can also be denoted as an *n*-dimensional vector \vec{SPM}_u , $\vec{SPM}_u = (s_1, \dots, s_n)$.

Identifying the severity of performance malfunctioning is another important aspect of ECU measurement. CPM is a set of parameters configured by users in representing the severity each performance malfunctioning feature could possibly cause. Define a function $p_{\langle user_x | SPM_u \rangle}$ representing the CPM configuration process for the user $user_x$ on the framework of discernment of use-case u , SPM_u . Denote this CPM as a n -dimensional vector $\bar{CPM}_u = (c_1, \dots, c_x, \dots, c_n)$, where c_x is the cost of damages when the x^{th} performance malfunction exists. Then, for $\forall s_x \in SPM$, $\exists c_x \in [0, +\infty)$ satisfying

$$p_{\langle user_x | SPM_u \rangle} : s_x \rightarrow c_x$$

It is users' responsibility to configure the CPM. In many cases, policy based automatic configurations are possible in order to make this process more friendly to users.

The third part of performance assessment is to find out the performance of the invoked system in terms of under-performing possibility when an application scenario is applied. The performance of the invoked system relates to the ability of physical resource that the system contains, and their managements. An n -tuple, PPM, is defined to measure the performance malfunctioning probability of a given function under a scenario. Denote s_x as a performance metric satisfying $s_x \in SPM_u$. For a use-case u , suppose the specified performance values as SPM_u' , where $SPM_u' = \{s_1', \dots, s_h'\}$. The scenario e is executed to apply the use-case u . p_x is defined as the possibility of performance malfunctioning that exhibited by the x^{th} performance metric, where $p_x = \Pr(s_x < s_x')$. Then, we have the probability of performance malfunctioning for scenario e , where $\bar{PPM}_e = (p_1, \dots, p_n)$. The PPM related to three factors: a) the ability and usability of components; b) importance of components to overall performance; and c) invoking frequency of components.

SPM and CPM can be given by the user, while PPM needs to be obtained from subsystems that compose the target Grid service.

3.2 Consequence Analysis for Performance Assessment

Consequence analysis investigates the dependency relation between the target Grid service and subsystems that composite the service. As mentioned above, multiple independent component Grid services will be invoked to deliver a Grid service to user. It is important to explicitly give the performance requirement of each component involved. Suppose there is a target Grid service A , it is expected to deliver a service to satisfy the performance requirement R . We denote the performance of the service A as $P(A)$, and require $P(A) \geq R$ to be satisfied in run time. The service A invokes a set of composite services, denote as $X = \{X_1, \dots, X_n\}$. The task of consequence analysis is to find out a set of performance requirements $R(X) = \{R_1, \dots, R_n\}$, for every composite service X_i , $X_i \in X$, satisfies $P(X_i) \geq R_i$, then $P(A) \geq R$ holds.

As mentioned above, there have dependency relations among services, where the satisfaction of one service's performance may relay on the satisfaction of other upstream services' performance. We then study how the performance requirement of individual composite service can be given. The back propagation method is used to

acquire the performance requirement of each composite service. The composite service that is closer to the user in workflow is analysed first for its performance requirement. Then, the performance requirement for the upstream service is generated. The upstream composite service will repeat the process until all composite services are reached.

For example, as shown in Figure 3, the service X_i extends X_j and X_k . When the performance of the composite service X_i is $P(X_i) \geq R(x_i)$ holds, the component X_i will issue $R(X_j)$ and $R(X_k)$ to its upstream composite services respectively in order to indicate the minimum requirement for upstream service. Then upstream composite service repeats the procedure until the leaf of the dependency tree has been reached.

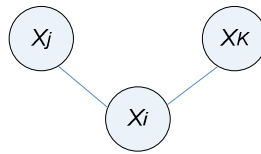


Figure 3. Dependency Relation of Composite Services

3.3 Subsystem Underperforming Risk Assessment

We consider a Grid service can be decomposed as many independent component Grid services, which actually contains software and hardware resources, and performs a set of functional activities. A set of invoking methods is pre-defined in response to requests of users. Components are self-managed and their performances can be individually instrumented. No dependency relation exists among components: whereas any performance degradation of one component does not directly result as performance degradation of any other components.

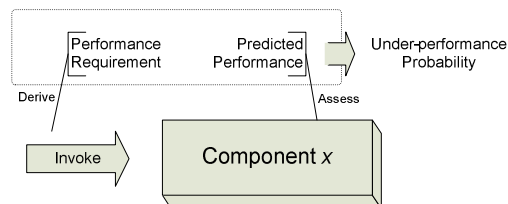


Figure 4. Performance Assessment Module for Component

The performance assessment process of a component is as shown in Figure 4. Following the definition of components, each of them is operating independently, and can be invoked through standard interfaces. When a component x is invoked by a scenario, performance requirements can be derived from the invoking request. The performance requirement is defined as the minimum performance of the component to guarantee the overall satisfaction of the composite Grid service. Assessment modules are used to predict the deliverable performance of the component x , which stands for the performance that the component x can possibly provide in response to the invoking request. The under-performance state exists once the performance of deliverable is under the requirement of scenario. We aim to obtain the under-performance probability in order to assess the performance of scenario.

A set of metrics is used to measure the performance of a composite service. Let a data structure $\langle x, y \rangle$ denote the y^{th} performance metric of the component x . Let performance metrics $R_{\langle x, y \rangle}(e)$ and $P_{\langle x, y \rangle}$ measure the performance requirement by scenario e and the delivered performance for $\langle x, y \rangle$. $u_{\langle x, y \rangle}$, $u_{\langle x, y \rangle} \in \{0, 1\}$, is the state value describing the healthy status of component x with regards to the performance metric $\langle x, y \rangle$, and satisfies

$$\begin{cases} u_{\langle x, y \rangle}(e) = 1, & \text{when } P_{\langle x, y \rangle} < R_{\langle x, y \rangle}(e) \\ u_{\langle x, y \rangle}(e) = 0, & \text{when } P_{\langle x, y \rangle} \geq R_{\langle x, y \rangle}(e) \end{cases}$$

An n -tuple, $U_x(e)$, is used to depict the performance status of component x under scenario e .

$$U_x(e) = (u_{\langle x, 1 \rangle}(e), \dots, u_{\langle x, n \rangle}(e))$$

where n is the total number of all measurable metrics for the component x .

The under-performing of components surely affect the performance of scenario. A performance malfunctioning assignment function is used to represent the degree of influence to SPM when a particular performance state exhibits in a particular component. A performance malfunctioning assignment function

$$m: SPM \rightarrow [0, 1]$$

is defined, when it verifies the following two formulas:

$$1) \forall A \in SPM, m_{U_x(e)}(A) \in [0, 1]$$

$$2) \sum_{A \in SPM} m_{U_x(e)}(A) \leq 1$$

where $m_{U_x(e)}$ denote the performance assignment function during the performance state $U_x(e)$. The higher value of $m_{U_x(e)}(A)$ denotes the higher influence of the state $U_x(e)$ to the performance malfunctioning A , and vice versa. The m function can be co-authored by system analysis and simulations. However, how to obtain the m function is not within the scope of this research. We consider the m function can be generated by a third party service.

We then study how to assess the performance of a component.

Let $R'_{\langle x, y \rangle}$ denotes the predicted performance for $\langle x, y \rangle$. For $\forall x, y$, there exist

$$v_{\langle x, y \rangle}(e) = \Pr(R'_{\langle x, y \rangle} < P_{\langle x, y \rangle}(e)).$$

Once the component x is invoked by scenario e , let $v_{\langle x, y \rangle}(e)$ be the degree of likeliness of under-performance measured from $\langle x, y \rangle$. The state of likeliness under-performance of component x can be represented as an n -tuple,

$$V_x(e) = (v_{\langle x, 1 \rangle}(e), \dots, v_{\langle x, n \rangle}(e)).$$

A similarity measurement function

$$\text{sim}: (X, Y) \rightarrow [0, 1]$$

is defined, where $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_n)$ are two n -tuples, and

$$\text{sim}(X, Y) = \prod_{k=1}^n (1 - |x_k - y_k|).$$

Then, when $V_x(e)$ can be obtained, the performance assignment function can be given as:

$$m_{V_x(e)}(A) = \sum_{U_x(e) \in \Phi_x} \text{sim}(U_x(e), V_x(e)) \times m_{U_x(e)}(A)$$

Then, we also can have

$$m_{V_x(e)}(\bar{SPM}) = \sum_{U_x(e) \in \Phi_x} \text{sim}(U_x(e), V_x(e)) \times m_{U_x(e)}(\bar{SPM})$$

From this formula, the relationship between the performance of component x and SPM is given.

3.4 Aggregate Performance Assessment

Multiple components are involved when a scenario is being executed. To account for the probability of a resource competition of component manifesting itself into user-end performance degradation, Dynamic Metrics (DM) is being used. Dynamic Metrics are used to measure the dynamic behaviour of a system based in the premise that active components are source of failure [2]. It is natural that a performance meltdown of component may not affect the remote instrumentation scenario's performance if not invoked. So, it is reasonable to use measurements obtained from executing the system models to perform performance analysis.

We study in the following of this section how can the value of Probability Performance Malfunctioning (PPM) of a scenario be given after the healthy measures of individual components are given. Beside the healthy statues of components, the frequency of invoking also affects the performance of scenario. Our methodology utilise a performance aggregation algorithm to provide the assessment of a scenario.

As components are invoked they become active for a specific duration performing the requested functionalities. Once the invoking requests exceed the limit of what the component can handle, some invoking requests will receive a response of under-performance. Therefore, there is a higher probability that, if an under-performance event is likely to exist in an active component, it will easily lead the scenario into malfunctioning.

A data structure denoting the Component Status Description (CSD) is defined as follows:

$$CSD_x(e) = \langle i, x, start_x(e), duration_x(e), m_{V(x)}(\bar{SPM}) \rangle$$

where i and x are the unified identification of invoking request and component; $start_x(e)$ and $duration_x(e)$ are, respectively, the start time and expected execution duration of component x . $start_x$ can be given by analysing the scenario that invokes the component. There are many technology can be found in literature analysing how to predict the execution duration of a task. For example, estimation of execution time can be done by using time stamps that are recorded in the simulation report. Therefore, we assume the value of $duration_x$ can be given by an external service.

When the CSD of all components that involve in a scenario are given, the component coordination can be depicted by a Time-Component Representation graph. As shown in figure 5(a), the dark lines denote the active state of the corresponding component. In order to conduct the performance combination, the component coordination relation needs to be mapped to a Discrete Time Representation. Discrete Time Representation describes the state of active components in a serial of discrete time fragments, which satisfies: a) no overlay between any time fragments; and b) the coordination relation of components keep unchanged in any time fragment. Figure 5

gives an example of mapping the Time-Component Representation of a scenario to Time-Discrete Representation.

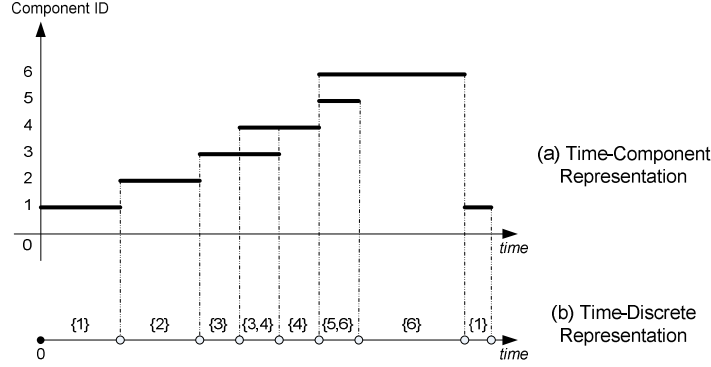


Figure 5. Component Coordinated Relation of Scenario

A data structure denoting the Scenario Status Description (SSD) is defined as follows:

$$SSD(e) = \langle j, \{x | x \in \text{set}(j)\}, \text{start}_j(e), \text{duration}_j(e), \bigcup_{x \in \text{set}(j)} m_{V(x)}(\bar{SPM}) \rangle$$

where j is the serial number for the component; $\text{set}(j)$ is the set of active components in the j^{th} time fragment; $\text{start}_j(e)$ and $\text{duration}_j(e)$ are the start time and the time length for the j^{th} time fragment when the scenario e is applied. Since all components are independently operated, the accumulated performance malfunctioning assignment can be given by the following formula:

$$\bigcup_{x \in \text{set}(j)} m_{V(x)}(\bar{SPM}) = 1 - \prod_{x \in \text{set}(j)} (1 - m_{V(x)}(\bar{SPM}))$$

Therefore, the PPM of a scenario e can be given as:

$$PPM(e, SPM_e) = 1 - \prod_{j \in \text{scenario } e} \left(1 - \frac{\text{duration}_j(e)}{T(e)} \cdot \bigcup_{x \in \text{set}(j)} m_{V(x)}(\bar{SPM})\right)$$

where $T(e)$ is total time required for scenario e .

Finally, the performance evaluation of a Grid service can be given as

$$\text{Cost}(e) = \sum_{A \in SPM} CPM(e, A) \cdot PPM(A)$$

where $\forall A \in SPM$,

$$PPM(e, A) = 1 - \prod_{j \in \text{scenario } e} \left(1 - \frac{\text{duration}_j(e)}{\sum_{j \in \text{scenario } e} \text{duration}_j(e)} \cdot \bigcup_{x \in \text{set}(j)} m_{V(x)}(A)\right), \text{ and}$$

$$\bigcup_{x \in \text{set}(j)} m_{V(x)}(A) = 1 - \prod_{x \in \text{set}(j)} (1 - m_{V(x)}(A)), \text{ and}$$

$$m_{V_x(e)}(A) = \sum_{U_x(e) \in \Phi_x} \text{sim}(U_x(e), V_x(e)) \times m_{U_x(e)}(A).$$

4. Showcase Scenario

We use a remote-telescoping application over Grid to showcase the performance assessment process. Figure 6 shows the workflow of the application. The user invokes the remote-telescoping service, while the service contains four encapsulated services in distributed geographical locations. Data Collection service contains telescoping units and gathers data. Raw Data Process service is in place to deal with raw data

passed by Data Collection service, while Knowledge Based is to be accessed during the raw data processing. Display Module gets post processing data and provides interactive graphic interface to users.

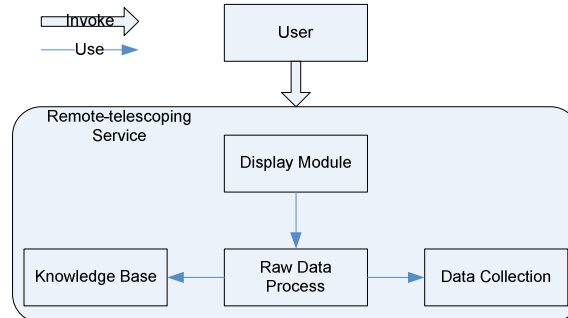


Figure 6. Workflow Model of Remote-telescoping Scenario

The SPM of the scenario can be given by a third party service. We denote the SPM of the remote instrumentation application as shown in Table 1. For the sake of simplicity and more clearly in showing the assessment process, we only list a small number of performance malfunctioning types. This table can nevertheless be extended to further depict more details of Grid services. When the SPM is obtained from a third party service, the CPM can be set by users. The value of CPM are also given in Table 1.

Malfunctioning type	Description	CPM
SPM_1	Image freeze	400
SPM_2	Display inconsistent	100
SPM_3	Data error	500
SPM_4	Stale data	200

Table 1. Performance Indicator Table

Therefore, the functional modules and their state space for under-performing are defined in Table 2. Components and the state space for under-performing are identified during the definition of the Grid service. m function can also be given by the analysis of history data and system analysis. In this showcase, we do not discuss how m assignment function is obtained, but assume the m assignment function value is given.

Components	Description	State Space for Under-performing	m (SPM_1)	m (SPM_2)	m (SPM_3)	m (SPM_4)
Data Collection (DC)	Collect data from the field	I: No response to req	0.8	0	0.1	0
		II: Data latency	0	0.4	0.4	0
		III: Data error	0.8	0	0.1	0
Data Process (DP)	Process raw data and output standard data format	I: CPU overuse	0.2	0.1	0	0.1
Knowledge Base (KB)	Provide related knowledge for data processing	I: Too many access	0	0.3	0.3	0.3
Display Module (DM)	Convert the standard data sheet into interactive graphic	I: Too many users	0.2	0	0	0.2

Trans. link 1 (TR1)	Data transmission between DC and DP	I: Overuse	0.1	0	0.1	0.1
Trans. link 2 (TR2)	Data transmission between KB and DP	I: Overuse	0	0.2	0.1	0
Trans. link 3 (TR3)	Data transmission between DP and DM	I: Overuse	0	0.2	0	0
Trans. link 4 (TR4)	Data Transmission to User	I: Overuse	0.2	0	0	0

Table 2. *m* Assignment Function for Modules in Remote-telescoping

Then, from the equation given in subsection 3.3, the value of the v -function can be given in the following table.

Component State	DC-I	DC-II	DC-III	DP-I	KB-I
v function	0.0001	0.0002	0.0001	0.005	0.005
Component State	DM-I	TR1-I	TR2-I	TR3-I	TR4-I
v function	0.005	0.01	0.01	0.01	0.02

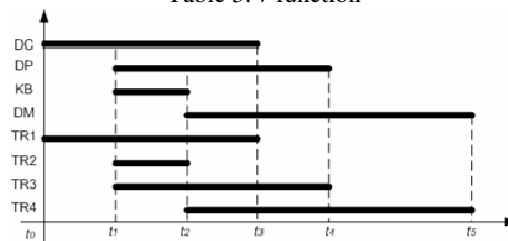
Table 3. v function

Figure 7. Activation Time Graph of Components

Furthermore, the activation relation of the components can also be obtained from the description of the Grid service, which is shown in Figure 7. Then the SSD can be given as:

- <1, {DC, TR₁}, t₀, 10, m₀>
- <2, {DP, KB, TR₂, TR₃}, t₁, 10, m₁>
- <3, {DC, DP, DM, TR₁, TR₃, TR₄}, t₂, 10, m₃>
- <4, {DP, DM, TR₃, TR₄}, t₃, 10, m₄>
- <5, {DM, TR₄}, t₄, 40, m₅>

We use the data in Figure 7 to construct a Dynamic Metric in analysing the performance. Then, the overall performance of the composite service can be calculated as shown in Table 4. Finally, the performance of the service can be measured.

SPM	CPM	PPM	Cost
1	400	0.0044	1.76
2	100	0.0014	0.14
3	500	0.0010	0.50
4	200	0.0014	0.28
Expected Cost of Underperforming (ECU):			2.68

Table 4. *PPM* measurements

5. Conclusion and Future Works

This paper describes the on-going research of the Grid service performance assessment. It explores the possibility of assessing the Grid service performance by using Guided Subsystem Approach. We prove that the approach is an effective way to measure the composite Grid services which include multiple loosely coupled component services. Top-down approach is used to capture the characteristics of user demands. Performance requirements are automatically translated and transferred to component services. Subsystem approach locally analyses the local available resources and demands, and concludes the possibility of under-performing for the local module. Dynamic metric is used to combine the performance assessment result together taking into account the importance of the component.

References:

1. Software Safety, NASA Technical Standard. NASA-STD-8719.13A, Sept. 1997, obtained from http://satc.gsfc.nasa.gov/assure/nss8719_13.html.
2. K. Moore, J. Cox, and S. Green, "Sonar - a network proximity service," Internet-Draft, <http://www.netlib.org/utk/projects/sonar/>, Feb. 1996.
3. P. Francis, "Host proximity service (hops)," Preprint, Aug. 1998.
4. P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, IDMaps: A Global Internet Host Distance Estimation Service, in IEEE/ACM Trans. on Networking, Oct. 2001.
5. S. Seshan, M. Stemm, R. Katz, "SPAND: Shared Passive Network Performance Discovery," USENIX Symposium on Internet Technologies and Systems, Dec 1997.
6. S. Bhattacharjee, Z. Fei, "A Novel Server Selection Technique for Improving the Response Time of a Relocated Service," Proc of IEEE Infocom, 1998
7. M. Crovella and R. Carter, "Dynamic Server Selection in the Internet," Proc of the 3rd IEEE workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'95), 1995.
8. Akamai Corporation. <http://www.akamai.com>

RAD Analysis of Adjusted Counter-Based Broadcast in MANETs

Sarah O. al-Humoud¹

Lewis M. Mackenzie¹

Mohamed Ould-Khaoua^{1,2}

Jamaldeen Abdulai¹

¹*Department of Computing Science
University of Glasgow
Glasgow, G12 8RZ, UK*

Email: {sara, lewis, mohamed, jamal}@dcs.gla.ac.uk

²*Department of Electrical & Computer Engineering
Sultan Qaboos University
Al-Khodh, 123, Muscat, Oman
Email: mok@squ.edu.om*

Abstract

Broadcasting is a vital operation in mobile ad hoc networks (MANETs) and it is crucial to enhance its efficiency to ensure successful deployment. Although flooding is ideal for broadcast operations due to its simplicity and high reachability it suffers from high packet collision which can degrade network performance severely. Counter-based broadcast schemes have been introduced to alleviate the limitations of flooding. This study introduces an enhancement to counter-based broadcast by adjusting the threshold value and the Random Assessment Delay (RAD) using minimal neighbourhood information. Simulation results of this study show that the new dynamic scheme provide good performance levels compared to the existing solutions.

1. Introduction

A MANET (Mobile Ad hoc NETwork) is an autonomous system consisting of a set of mobile nodes that are free to move without the need for a wired backbone or a fixed base station. MANETs have several marked characteristics linked to their lack of a centralized infrastructure. First, MANETs are decentralized, with all mobile nodes functioning as routers and all wireless devices being interconnected to one another. Second, a MANET has a dynamic topology. Nodes are free to roam in or out of the geographical coverage area, causing rapid and unpredictable changes to the network topology over time [1]. Thirdly, a MANET operates on bandwidth constrained variable-capacity wireless links. Wireless communication is typically subject to frequent disconnections, low network throughput, high response time and lack of security [2, 3]. Fourthly, a MANET is often bounded by energy-constrained operations as nodes are often hand-held and battery-powered wireless transmitters [3]. MANETs applications cover many areas that share a common need of an infrastructureless rapid deployment network, examples being emergency operations, collaborative and distributed computing, inter-vehicle communication, Hybrid wireless networks, and military applications.

Broadcasting is the process by which one node sends a packet to all other nodes in the network. Broadcasting has been extensively used in networks for applications such as discovering neighbours, collecting global information, naming, addressing, and sometimes helping in multicasting [4]. Particularly in a MANET, due to node mobility, broadcasting is expected to be performed more recurrently. For example, broadcast service could be used for paging a particular host, sending an alarm signal, and finding a route to a particular host [5]. Moreover, Ad hoc On-Demand Distance Vector Routing (AODV)[6], Dynamic Source Routing (DSR)[7], Zone Routing Protocol (ZRP)[8], and Location Aided Routing (LAR)[9] are some examples of routing protocols that rely on broadcasting for route discovery.

Blind flooding is the basic approach to broadcasting where every node in the network forwards the received packet exactly once. Blind flooding is simple and guarantees high reachability, but at the expense of inefficient utilisation of system resources such as channel bandwidth and battery power of mobile nodes. The approach is associated with high redundant

transmissions that can cause high channel contention and packet collisions in the network. This phenomenon of blind flooding is referred to in the literature as *broadcast storm problem* [5].

Several methods have been proposed to alleviate the broadcast storm problem associated with blind flooding [5]. However, these methods can be classified into two categories. The first category of broadcast schemes is referred to as non-deterministic broadcast schemes. These schemes mitigate the network congestion levels by reducing the number of retransmitting nodes. This is achieved by inhibiting some intermediate nodes from forwarding the received broadcast packets using some local topological characteristics. Examples of the non-deterministic broadcast schemes include counter-based, area-based, distance-based, and probability-based schemes [5].

In counter-based scheme, a node is allowed to forward a broadcast packet if the number of duplicate packets received is less than a threshold value. The area-based scheme allows only the nodes that have the potential of reaching more nodes which have not been covered by the broadcast area to forward the packet. Furthermore, in the distance based scheme the forwarding node compares the distance between itself and each neighbouring node that has previously rebroadcast the packet against a threshold distance value. If the distance from any neighbouring node is greater than a threshold, the node refrains from forwarding the packet. A node using probabilistic broadcast scheme rebroadcasts a received packet with a predetermined probability $p < 1$. The scheme is similar to blind flooding if the predetermined probability is 100%.

The second category of broadcast schemes predetermines a set of forwarding nodes based on global topological information of the network. These schemes are referred to as deterministic broadcast schemes. Examples of deterministic broadcast schemes include pruning [10], multipoint relaying [11], node-forwarding [12], neighbour elimination [13], and clustering [14].

In general, in non-deterministic broadcast schemes, nodes make instantaneous local decisions about whether to broadcast a message or not using information derived only from overheard broadcast messages. Consequently, non-deterministic schemes incur a small communication overhead and can adapt to changing environments when compared to deterministic schemes [15].

In this study we propose a new efficient counter-based broadcast scheme that aims to reduce the broadcast storm problem without degrading the reachability. Our new broadcast scheme dynamically adjusts the counter threshold at a forwarding node based on its local topological characteristics. Our simulation results reveal that the proposed scheme can achieve better performance in terms of saved rebroadcast while providing comparable reachability when compared against the traditional counter-based scheme and the blind flooding based broadcast.

The rest of this paper is organised as follows. Section 2 will explain the related work of the counter-based rebroadcast. Section 3 outlines our proposed adjusted-counter-based scheme. Section 4 presents the simulation results of the proposed algorithm. Finally, section 5 is our future directions and conclusion.

2. RELATED WORK

Counter-based broadcasting was initially proposed in [16] as a mechanism to reduce redundant rebroadcast packets and alleviate the problems associated with blind flooding. The basic idea of the counter-based scheme is based on the inverse relation between the *expected additional coverage* (EAC) and number of duplicate broadcast packets received [5,16].

Essentially, if the number of duplicate packets received at a node high then, the EAC is expected to be low if the packet forwarded. This is because most if not all of the neighbours of the forwarding node would have received the packet.

A node is prevented from forwarding a received broadcast packet when the EAC at the node is low. The idea of EAC is depicted by the example shown in Figure 1. The hollow shaped nodes are source nodes that initiate the broadcast transmission and the solid black nodes are nodes we use to clarify our idea; we will refer to them as *black-a* and *black-b*. Apparently, the neighbourhood density of black-a is higher than black-b. Therefore, the number of duplicate broadcast packets that would be received by black-a is higher. Moreover, it is likely that most of the nodes within the transmission range of black-a would already have been reached by other forwarding nodes. Therefore, the EAC of black-a is lower than the EAC of black-b.

The counter-based broadcasting scheme works as follows: when receiving a message for the first time a counter c is initiated to keep track of the number of duplicate packets received and a *random assessment delay* (RAD) timer is also initiated. The RAD is a jitter randomly chosen between 0 and T_{max} seconds. This delay is necessary for two reasons. First, it allows nodes adequate time to receive redundant packets and assess whether to rebroadcast. Second, the randomized scheduling prevents collisions [17]. As soon as the RAD timer expires the counter c is compared against a fixed threshold value C ; broadcast is inhibited if $c \geq C$. An adaptive counter-based scheme was proposed in [4]. The authors have suggested extending the traditional fixed counter threshold scheme to incorporate the number of neighbours at a node. Specifically, the decision to forward the broadcast packet is determined by the function $C(n)$ where n is the number of neighbours of the forwarding node. However, they have stated that the function $C(n)$ is undefined [4].

$$RAD = [0, T_{max}) \quad (1)$$

Other variants of the counter-based broadcast scheme include color-based [18] scheme and the distance-aware counter-based scheme [19]. The main idea of the first scheme is to assign colours to the broadcast packets. Using η different colors C_1, C_2, \dots, C_η each forwarding node selects a colour which it writes to the colour-field of the broadcast packet. All nodes which hear the packet rebroadcast it unless they have heard all η colours by the time a random timer expires. The question that could be asked is: what if $\eta=3$ and a node received 2η messages having the colours $\{C_1, C_2\}$ only? According to color-based broadcasting, this node will still rebroadcast the message although it received a high number of messages, six. Additionally, the proposed color-based broadcasting scheme suffers from the same drawback that the fixed counter-based suffers from: it scores high reachability only when used with homogeneous density networks; when the network is sparse $\eta=3$, and when dense $\eta=2$. Moreover, the authors stated that by increasing η reachability increases. However, they also claimed that there is no such threshold value that can provide full-reachability for any arbitrary connected network. The distance-aware counter-based broadcast is based on the counter-based algorithm proposed by Ni et al [5]. Additionally, this algorithm introduces the concept of distance into the counter-based broadcast scheme by giving nodes closer to the node transmission range border a higher rebroadcast probability since they create better EAC.

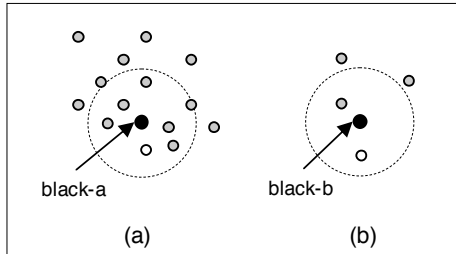


Figure 1: Expected Additional Coverage example

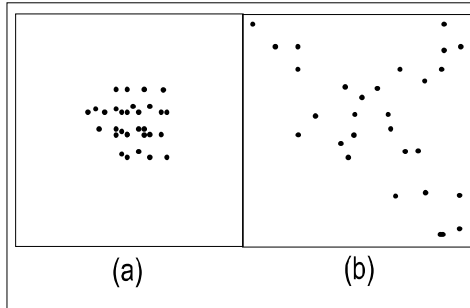


Figure 3: Example of changeable network topology. (a) Dense network with 30 nodes (b) the same network with nodes forming several sets of sparse networks.

ACB_Broadcast_Algorithm

Pre: **avg** is average number of neighbours
a broadcast packet **m** at node **X** is heard

Post: rebroadcast the packet or drop it, according to the algorithm

```

1.  Get the Broadcast ID
2.  Get degree n of node X
3.  If n < avg then
    Sparse network
    C = C1
    Tmax = x/RF1
4.  Else
    Dense network
    C = C2;
    Tmax = x/RF2
5.  End if
6.  Set RAD
7.  c = 1
8.  While (RAD) Do
    If (same packet heard)
        Increment c
9.  End while
10. If (C < c)
    drop packet
    exit algorithm
11. End If
12. Submit the packet for transmission
End ACB_Broadcast_Algorithm

```

Figure 2: Adjusted counter-based broadcast algorithm

3. ADJUSTED COUNTER-BASED BROADCAST

Existing counter-based broadcasting schemes use a fixed threshold value to alleviate the shortcomings of pure flooding; however, we have the following remarks on counter-based broadcast schemes with existing fixed threshold value. First, the topology of MANETs is often random and dynamic with varying degrees of node density in various regions of the network as shown in Figure 3. The network may contain sparse and dense regions. Therefore, fixed counter threshold approach suffers from unfair distribution of C since every node is assigned the same value of C regardless of its local topological characteristics. Second, there exist a trade-off between reachability and saved rebroadcast. While using small threshold values provides significant broadcast savings, unfortunately, the reachability will degrade sharply in a sparse

$$T_{max} = x / RF \quad (2)$$

network. Increasing the value of C will improve the reachability, but, once again, the amount of

saving will be sacrificed [4]. Third, according to my knowledge, there is no proposed method that dynamically and autonomously changes the counter threshold value.

Accordingly, sparse networks need a higher chance to rebroadcast than dense networks. This could be achieved by one of two ways or a combination of them. First, altering the threshold value C to adapt to network density where a small threshold value C_2 is used for dense networks (high n) and a large threshold value C_1 for sparse networks (low n). Second, altering the *Random Assessment Delay* (RAD) where a small RAD is used for dense networks (high n) and a large RAD for sparse networks (low n). Moreover, a *Random Factor* (RF) is introduced as shown in Equation 2 where x is a random number between zero and one. Equation 2 implies the inverse relation between RF and RAD by substituting T_{\max} in Equation 1.

The adjusted counter-based broadcast algorithm, Figure 2 works as follows: when receiving a broadcast message for the first time a node sets the RAD, which is randomly chosen between 0 and 1 second and initiates the counter to one. Following, the node checks the number of neighbours n against the average number of neighbours avg ; if $n < avg$ then the network is considered sparse and C_1 is selected as the threshold value and RF is set to RF_1 , otherwise the threshold value is set to C_2 and RF is set to RF_2 . Additionally, the values C_1 and C_2 are selected in a way that considers the expected additional coverage EAC. That is, c_1 (sparse network threshold) should be in a way larger than c_2 (dense network threshold) in order for the node to have a higher chance to rebroadcast in a sparse area whilst the EAC of the sparse network is higher than that of the dense network as we mentioned with an example previously. The same principle applies to RAD, that is, RF_1 is selected to be smaller than RF_2 . After selecting the threshold value and during the RAD, the counter is incremented by one for each redundant packet received. When the RAD expires the counter is checked against the threshold value, if the counter is less than or equal to the threshold, the packet is rebroadcast. Otherwise, it is simply dropped.

While blind flooding ensures that every node in the network receives the broadcast packet (i.e. high reachability) at the cost of high communication overhead (i.e. low save rebroadcast), our proposed scheme aims at significantly reducing the communication overhead while still achieving comparable reachability when compared to blind flooding. To achieve this, our broadcast approach utilizes neighbourhood information, i.e. number of neighbours in particular to select the best counter threshold. The *number of surrounding neighbours* (n) a node have is known by periodic exchange of HELLO packets among neighbouring nodes.

4. PERFORMANCE ANALYSIS

We evaluate the performance of our proposed algorithm using the ns-2 network simulator [20]. Ns-2 is a discrete event simulator targeted at networking research for both wired and wireless networks. Moreover, ns2 has been used by most researchers for performance evaluation in MANETs research [17,19,18]. The present study investigates the performance impact of system parameters on the proposed algorithms; notably node mobility, network density and traffic load. In addition, we have investigated the effects of RAD RF values on the performance of the proposed algorithm. For system parameter under investigation, the counter-threshold values have been fixed $c_1 = 2$ and $c_2 = 3$ and the RF values (RF_1 , RF_2) are varied

over the range (100, 10), (100, 1) and (10, 1). We compared the results of ACBase against the traditional counter-based with counter threshold set to 2 [17] and blind flooding.

4.1. *Simulation Setup*

The radio propagation model used in this study is the Ns-2 default, which uses characteristic similar to a commercial radio interface, Lucent's WaveLAN card with a 2Mbps bit rate [21]. The Distributed Coordination Function (DCF) of the IEEE 802.11 protocol [22] is used as the MAC layer protocol. The mobility model uses the random waypoint model in a flat area of 1500m x 500m. In a random waypoint mobility model, each node at the beginning of the simulation remains stationary for a pause time seconds, then chooses a random destination and starts moving towards it with a randomly selected speed from a uniform distribution [0, max-speed]. After the node reaches its destination, it again stops for a pause-time interval and chooses a new destination and speed. The simulation is allowed to run for 100 seconds for each simulation scenario. The following simulation assumptions are commonly used in many MANET studies [5,17]. We assume that a host can detect duplicate broadcast messages. This is essential to prevent endless flooding of a message. One way to do so is to associate with each broadcast message the broadcast ID [4,5]. Moreover, we assume that nodes have sufficient power supply to function properly throughout the simulation time.

Table 1 shows some of the essential simulation parameters that have been used in the evaluation of our protocols.

TABLE 1: SIMULATION PARAMETERS

Simulation parameter	Value
Simulator	ns-2 (version 2.33)
Network Area	1500x500 m ²
Transmission range	250 meters
Simulation Time	100 sec
Number of Trials	10
MAC layer protocol	IEEE 802.11
Packet Rate	2 packets per sec per node
Confidence interval	95%

4.2. *Performance measures*

Below is the performance metrics used to evaluate the performance of the proposed broadcast approach:

- *Reachability* (RE), defined as r/e , where r is the number of hosts receiving the broadcast packet and e is the number of mobile hosts that are reachable, directly or indirectly, from the source host.
- *Saved Rebroadcast* (SRB), defined as $(r - t)/r$, where r is the number of hosts receiving the broadcast message, and t is the number of hosts that actually transmitted the message.

- *Average latency* (Delay), which is the interval from the time the broadcast, was initiated to the time the last host finished its rebroadcasting.

4.3. *Simulation and Discussion*

To analyze the impact of different RAD values on the performance of our proposed Adjusted Counter Based approach (ACBase), we have divided our study into three parts: first the study of node mobility, then density and finally traffic load. In each part we examine the impact of different RAD values fixing the threshold to (2x3) and (2) for ACBase and Cbase respectively. The results for blind flooding have been added for the sake of completeness

4.3.1. Effects of Mobility and RAD

We investigate the effects of mobility on the performance of the proposed algorithm by varying the maximum nodal speed over a range of 1, 5, 10, 15, and 20 m/sec. The number of nodes deployed over the area of 1500m x 100m had been fixed at 50. Ten nodes were randomly selected to initiate the broadcast process. Each node sends 2 packets/sec. Packet size of 512 bytes has been used. Figure 4a depicts the SRB versus maximum nodal speed. As can be shown in the figure ACBase can achieve high SRB when compared against the counter-based and blind flooding when the mobility of nodes is increased from low to medium mobility. For example, the ACBase SRB with RFs of (100, 1) is around 35% and that of counter-based is around 17% at low mobility of 1m/sec. At medium to high mobility (i.e. from 10m/sec) the SRB of ACBase is around 25% and that of counter-based is 10%. In addition, apparently the SRB values for ACBase increases with increasing the waiting time factor (RF). For instance, the SRB of ACBase with RFs of (100, 1) is about 3% higher than that of ACBase with RFs of (100, 10).

Figure 4b shows reachability versus mobility. All the algorithms present similar trends of reachability for all node speeds. However, the reachability is low (i.e. between 90 % and 95%) when nodes mobility is low. This is due to poor connectivity in low mobility environments. In a mobility environment such as 20 m/sec, the reachability for all the algorithms is around 100%

Figure 4c investigates the effects of mobility and random delay factor on average latency (delay for short) of the protocols. The figure shows the ACBase approach is outperformed by both counter-based scheme and blind flooding for mobility scenarios. But the delay for all the protocols remains fairly constant across all mobility. The figure also reveals that the delay incurred by ACBase is worsened when the RFs decrease from (100, 10) to (100, 1).

4.3.2. Effects of Density and RAD:

This section evaluates the effects of node density and random delay factor on the performance of the proposed protocol. In this study we vary the density by increasing the number of nodes deployed over a fixed area of 1500m x 500m. The number of nodes has been varied from 25 to 200 in steps of 25 nodes with each node moving at a speed between 0 and 5 m/sec. To reduce effects of traffic load, one node was randomly selected to initiate the broadcast process at a sending rate of 2 packets/sec.

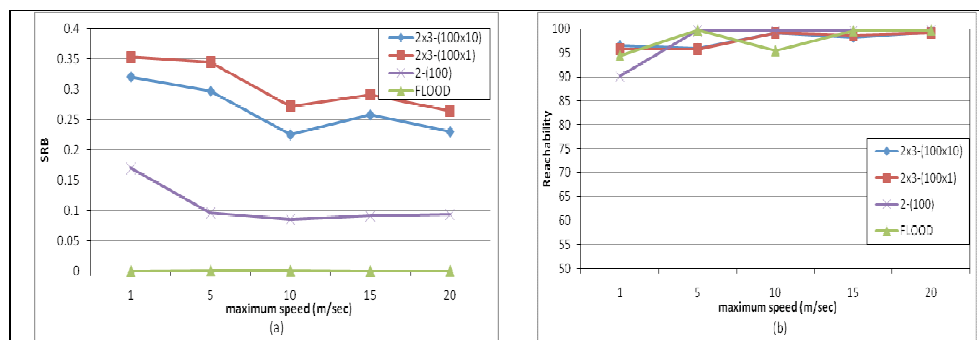
Figure 5a presents SRB versus network density. The SRB of ACBase increases with increasing density. However, the SRB of counter-based and blind flooding remains almost flat with increasing node density. This is due to the factor that the counter-based scheme uses a fixed counter value and fixed random factor for all the regions in the network. However, a node

using ACBase set these values low when in dense region and high when in a sparse region of the network. At low density of 25 nodes, the ACBase with RFs of (100, 1) and the counter-based scheme achieves similar SRB of around 10%. But at high density of 200 nodes, the ACBase with RFs of (100, 1) achieves superior performance of SRB reaching about 60%.

Figure 5b shows the effects of network density on reachability. All the algorithms present similar trends of reachability with increasing network density. The reachability increases almost linearly from low to medium network density and reaching 100% at high network density. The poor reachability at low network density is due to poor connectivity suffered by sparse networks. In figure 5c we present results of the effects of density and random factor on average latency. ACBase with RFs of (100, 10) achieves comparable performance in terms of delay with the counter-based and blind flooding across all network densities. However, the delay incurred by ACBase with RFs of (100, 1) is much high, reaching about 500% of the delay incurred by the counter-based scheme.

4.3.3. Effects of Traffic Load RAD:

Studying network behaviour under unfixed traffic load, 50 nodes was considered. Moreover, number of sending nodes employed was 1, 5, 10, 15, 20, and 25 nodes. Given that each of the sending nodes has the chance to send 2 packets/ sec, the total network load would be: 2, 10, 20, 30, 40, and 50 packets/ sec. Figure 6a shows ACBase SRB gain over Cbase of 144% with an average performance of 24% and 12% for ACBase and Cbase respectively. With respect to reachability the performance was quite comparable with a gain of 1.6% for ACBase against Cbase. However, reachability degrades as the traffic load increases, that is due to higher collisions and contentions that worsen reachability. The same is to say with regards to delay that is affected negatively with higher traffic loads. Moreover Delay gain is 9% of ACBase over Cbase.



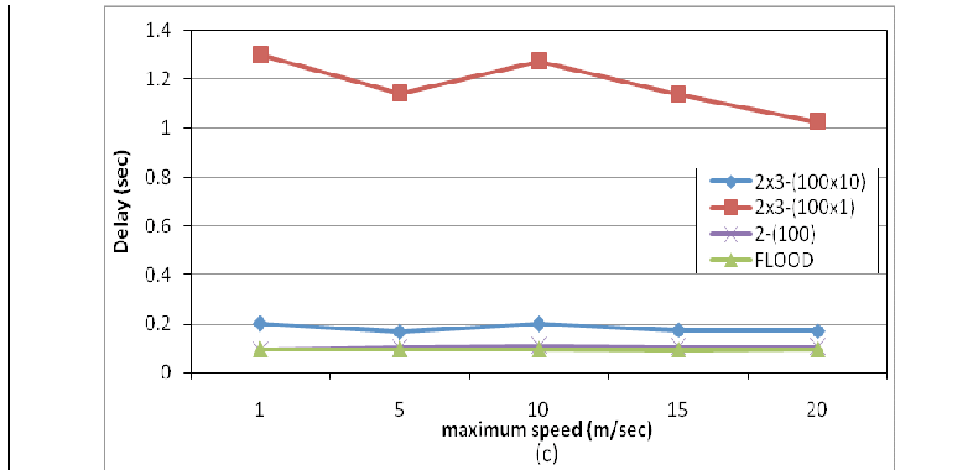


Figure 4. Effects of node mobility and random assessment delay (RAD) on the performance of the protocols for 100 nodes over a topology area of 1500m x 1500m: (a) Saved Rebroadcast (SRB): (b) Reachability and (c) Delay

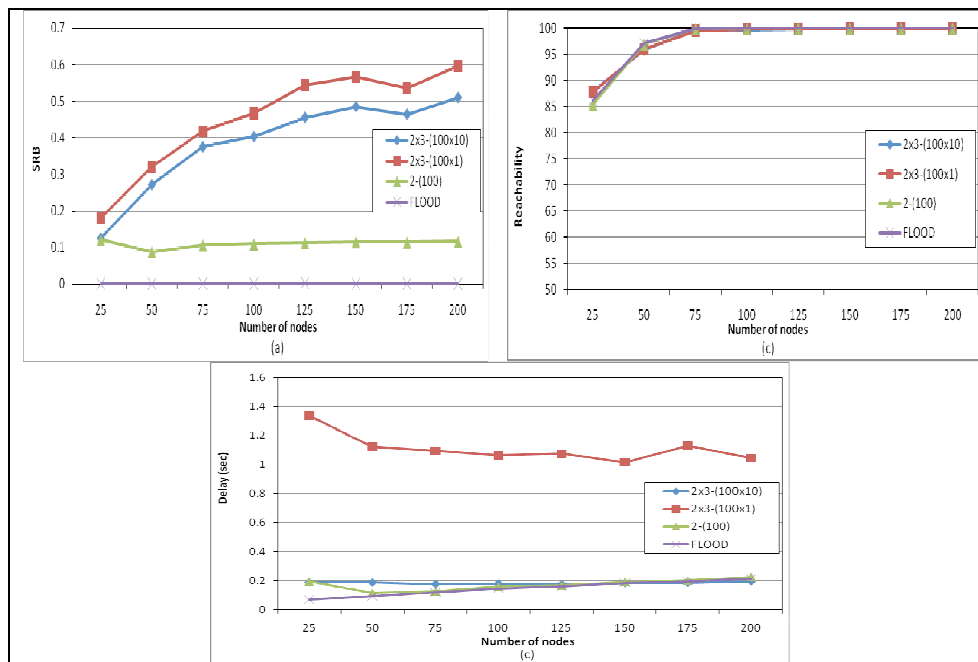


Figure 5. Effects of network density and random assessment delay (RAD) on the performance of the protocols for a traffic rate of 2 packets/sec with 512 bytes packet size: (a) Saved Rebroadcast (SRB): (b) Reachability and (c) Delay

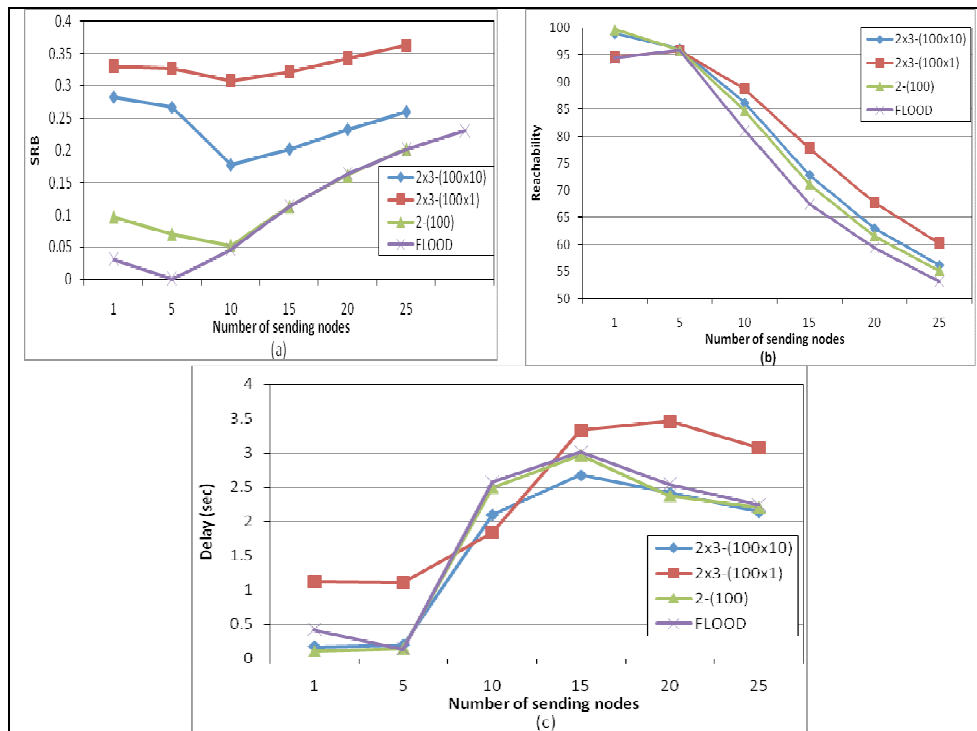


Figure 6. Effects of network traffic load and random assessment delay (RAD) on the performance of the protocols for 50 nodes over topology area of 1500m x 500m: (a) Saved Rebroadcast (SRB); (b) Reachability and (c) Delay

5. CONCLUSION AND FUTURE WORK

This paper has analysed the impact of variable RAD values on the performance of the proposed Adjusted Counter-Based broadcasting scheme in MANETs. We analysed our algorithm under three different variations: mobility, density, and traffic load. Moreover, the effect of alternative RAD values on SRB, Reachability and delay was investigated. ACBase broadcasting scheme scored a large gain in SRB compared to the fixed counter-based with a similar reachability and a slight loss in delay. As a continuation to this work, we plan to investigate the impact of different threshold values on the performance of the new ACBase scheme.

REFERENCES

- 1 C. S. R. Murthy and B. S. Manoj, "Ad Hoc Wireless Networks: Architectures and Protocols," Prentice Hall PTR, 2004.
- 2 D. Katsaros, A. Napoulos and Y. Manolopoulos, "Wireless Information Highways," IRM Press, 2005.

-
- 3 S. Corson and J. Macker, "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," 1999.
 - 4 Y.-C. Tseng, S. Member, S.-Y. Ni, and E.-Y. Shih, "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network," IEEE TRANSACTIONS ON COMPUTERS, vol. 52, 2003
 - 5 Y.-C. TSENG, S.-Y. NI, Y.-S. CHEN and J.-P. SHEU, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," Wireless Networks 8, pp. 153–167, 2002
 - 6 C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," MILCOM '97 panel on Ad Hoc Networks, 1997.
 - 7 D. B. Johnson, D. A. Maltz, and J. Broch, "Dynamic Source Routing in Ad Hoc Wireless Networks," in Mobile Computing. vol. 353, I. a. Korth, Ed., 1996.
 - 8 Z. Haas. "A new routing protocol for the reconfigurable wireless networks," Proc. of the IEEE Int. Conf. on Universal Personal Communications. 1997.
 - 9 Y.-B. Ko, and N.H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom_98), 1998.
 - 10 J. Wu and F. Dai, "Broadcasting in Ad Hoc Networks Based on Self-Pruning," IEEE INFOCOM, 2003.
 - 11 A. Qayyum, L. Viennot and A. Laouiti, "Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks," the 35th Annual Hawaii International Conference on System Sciences, 2002.
 - 12 P.-J. W. Gruia Calinescu, Ion I. Mandoiu and A. Z. Zelikovsky, "Selecting forwarding neighbors in wireless ad hoc networks," 2004.
 - 13 I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," IEEE, pp. 14–25, 2002.
 - 14 W. Lou and J. Wu, "A cluster-based backbone infrastructure for broadcasting in manets," IPDPS '03. IEEE Computer Society, 2003.
 - 15 P. Rogers and N. Abu-Ghazaleh, "Towards reliable network wide broadcast in mobile ad hoc networks," 2004.
 - 16 S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network", Proc. Mobicom'99, 1999.
 - 17 W. Brad and C. Tracy, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing. 2002
 - 18 A. Keshavarz-Haddad, V. Ribeiro and R. Riedi, "Color-Based Broadcasting for Ad Hoc Networks," IEEE, 2006
 - 19 C. Chen, C.-K. Hsu, and H.-K. Wang, "A distance-aware counter-based broadcast scheme for wireless ad hoc networks," Military Communications Conference. IEEE, vol. 2, pp. 1052- 1058, 2005
 - 20 The Network Simulator ns-2: www.isi.edu/nsnam/ns/
 - 21 "IEEE802.11 WaveLAN PC Card - User's Guid," pp. A-1
 - 22 IEEE Standards Department. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE standard 802.11-1997, 1997

Simulation of a tandem router system for mobile network traffic with different source traffic distributions

Aisha Khalid Khan*

Abstract

The aim of this paper is to analyze the performance of a simple wireless router network of mobile hosts with different types of source traffic distributions using discrete event simulations. The work presented here builds on an existing model of a wireless router network developed by Gulpinar et al [1] which consists of two finite capacity queueing nodes connected in tandem, with Poisson arrivals at the source. The focus of this paper is on finding out how the MRT varies with changing source traffic distributions. We begin with experiments using Poisson arrivals. Although the Poisson assumption makes the problem analytically tractable, it does not accurately represent the correlated nature of network traffic. As a next step in the quest to model more realistic source traffic, a heavy tailed distribution i.e., the (truncated) Pareto distribution is used for inter-arrival times of packets generated by the source. With truncated Pareto distributed arrivals, the undesirable effects (high mean and variance of response time, under-utilisation of servers and high loss rate in particular) of heavy tails on performance measures become apparent owing to the bursty nature of arrivals. It is also observed that while Poisson arrivals maintain their exponential features by the time they reach the receiver, the power law characteristics of Pareto arrivals are lost. The effect of batch arrivals is also considered in which customers arrive at the queue in form of fixed size batches. Finally, source arrivals with exponentially distributed inter-arrival times and (truncated) Pareto distributed batch sizes are modelled which show drastic results in the form of very high packet loss and extremely low server utilisation. The paper also discusses how the trade-off between lowest possible packet loss and high variability of response time is based on the choice of the ratio of arrival to recovery buffer sizes.

1 Introduction

Owing to the continuous growth of the information exchange industry and the desire of Internet users to have ubiquitous access, the next generation of networking systems will be data-centric and mobile. However, wireless networking brings with it various challenges. The foremost of these is the dynamic link organization. A mobile device transmitting and receiving radio signals exposes wireless transmission to the effects of noise and interference from other devices. A network node can get isolated as it cannot reach any of its neighbours due to interference or because none of the neighbouring nodes is within the node's radio range. This depends directly on the buffering capabilities of each node [2]. The connectivity of the node, i.e., the number of neighbours that it can reach is directly related to retransmission rates [2]. Another problem is the high variability in QoS requirements such as error rate, latency and bandwidth, caused by mobility and

*Computer Science Department, MCS, NUST, Rawalpindi, Pakistan, aisha.khan@mcs.edu.pk

unpredictable link variation. A mobile device has another limitation in the form of limited battery life, and therefore, efficient energy consumption by the mobile device becomes an issue to be addressed when dealing with wireless networks.

The research in [1] attempts to deal with the issue of mobile routers by presenting an “analytic, static, optimization-based design of routers for use in wireless communications”. The model consists of a network of two mobile nodes connected in tandem. At each node, which is a first come first served (FCFS) queueing node, there are arrival (for queueing) and recovery buffers. It is assumed that the packets arrive at the server as a Poisson process. The packets are served when they reach the front of the arrival queue. Service time is assumed to be an exponential random variable.

The buffers at queueing nodes are of finite capacity. Hence, the ratio of arrival buffer size to recovery buffer size is critical to optimal performance. The work in [2] further extends the model to have generally distributed service times and batch arrivals which are still exponentially distributed. In both these studies, the main aim was to find the optimal ratio of arrival and recovery buffers that minimizes end to-end transmission time. While the Poisson model very accurately represents voice traffic, the same is not true in case of data networks. Extensive studies of modern data networks [5-8] have shown that the traffic is bursty and the simple Poisson assumption for source traffic is no longer sufficient. In this paper, we have made an effort to model source traffic more realistically by changing the statistical distribution of the arrival process by extending the discrete event simulation used in [2].

The rest of the paper is organized as follows. Section 2 discusses the model in detail. In section 3, we have presented the results of simulations of various traffic types including M/M/1 and TP/M/1 queues. We have only presented results for the Mean Response Time [MRT] in each case. Section 4 discusses how the model has been extended to allow truncated Pareto distributed batch sizes and how the MRT changes in this case. We also compare the results of all the different source traffic distributions and analyze them. Section 5 concludes the paper and section 6 discusses some future directions in which this research can evolve.

2 Model Design

We consider a two-node, tandem network of mobile routers as shown in Figure 2.1 taken from [1]. The sender (source) is connected to Router1 which is a FCFS queueing node. Router1 has an arrival (for queueing) buffer of size A_1 and a recovery buffer of size R_1 . Router1 is connected in tandem to another queueing node Router2 which has its arrival and recovery buffers of size A_2 and R_2 respectively. The total buffer capacity C_i at each node i is finite and fixed. Only the sizes of the arrival and recovery buffers can be changed keeping the total capacity the same. Hence,

$$C_i = A_i + R_i \quad \text{for } i = 1, 2$$

If R_i is set too small, transmission times will be large due to the need for more retransmissions from the sender. If it is too large, the arrival buffers will be too small and so transmission times will again be large due to losses and the resulting time-outs and retransmissions. It is an optimisation exercise to find the best ratio of recovery buffer to arrival buffer sizes as has been done in [1] and [2]. The motivation for keeping the buffer capacity fixed and trying rigorously to find the optimal ratio of arrival to recovery buffer sizes is to make more efficient use of fast memory in routers. So, instead of adding more

buffer capacity, we can change the ratio of arrival to recovery buffer sizes to suit our application.

The packets sent by the sender node are stored in the arrival buffer A_1 of Router1 (if the buffer has enough capacity) and are served when they reach the front of the queue. The recovery buffer at each node is used to store any packet which has been successfully transmitted downstream. If a recovery buffer is full on transmitting a packet successfully, then the oldest packet is deleted i.e., it works in a cyclic round robin fashion. The service time at each of the two nodes comprises the time to look up router tables, manage header information and dispatch the packet so that the next can be processed. From Router1, the packets are sent on to Router2 which in turn routes the packets to the receiver node. A packet that reaches the receiver without getting corrupted (i.e., a clean packet) is successful. Any packet which has not been transferred successfully is lost. Losses occur at a queueing node either because packets are rejected when they arrive at a full arrival buffer or when corrupted packets are detected on receipt at a router. In the former case, a full retransmission is initiated by the source after a timeout period which causes an additional transmission delay. Whether or not the retransmission is to be attempted is determined by the resend probability. In the latter case, a negative acknowledgement (N-ACK) is sent to the upstream node. If the upstream node's recovery buffer has a clean copy of the packet, the packet is re-transmitted; otherwise it is deemed lost forever. Hence, the model takes into account the randomness in traffic and errors, both of which are important aspects to be considered while dealing with wireless networks. In the model, it is assumed that a packet with corrupted data is not buffered.

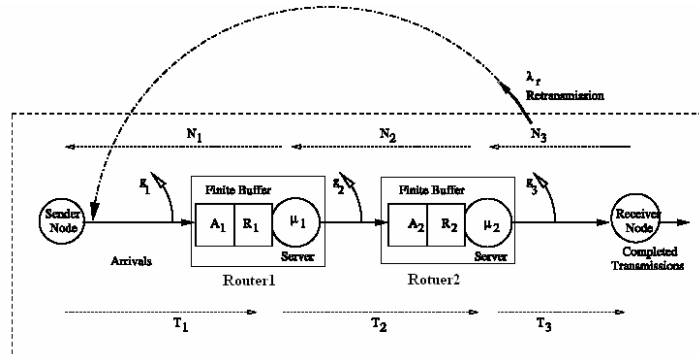


Fig 2.1 A two-node tandem router network (taken from [1])

2.1 Model Structure and Parameters

The arrival process has a mean arrival rate λ . Packet service times, which are independent of each other and of the arrival process, have mean values $1/\mu_1$ and $1/\mu_2$, respectively; i.e. the service rates are μ_1 and μ_2 . The probability that a packet is lost because of corruption that results in garbage data arriving at R_1 , R_2 and the receiver are g_1 , g_2 and g_3 respectively. Such garbage is detected on arrival at a node before it enters the queue and

leads to a N-ACK being sent upstream. N_1 , N_2 and N_3 are the N-ACK delays from Router1 to Sender, from Router2 to Router1, and from Receiver to Router2 respectively. The constant communication delays for a packet passing over the links between the Sender and Router1, Router1 and Router2, and Router2 and Receiver are T_1 , T_2 and T_3 respectively.

The successful transmission of an uncorrupted packet is recognized by the sender on receipt of a positive acknowledgment (i.e., an ACK) from the receiver according to the transmission protocol. The sender sets a timer when it transmits a packet and if no acknowledgment is received when the timer reaches the predefined time-out value *TimeOut*, the packet is considered to have been lost and a retransmission may be attempted. The probability of retransmission (*ResendProb*) is also variable. When *ResendProb* = 1, there is always a retransmission attempt causing congestion resulting in higher transmission delays. When *ResendProb* = 0, no retransmissions are allowed so losses are higher but congestion is lower. Hence, successful transmissions are, on average, faster.

The motivation for introducing loss rates and corruption probabilities comes from the notion of mobility. Mobile networks are subject to topology changes due to movement of the nodes. Also, since the communication uses air as the medium, interference from other devices and noise in the communication channel also affect reliable transmission. Therefore it is important to accommodate these error probabilities in the model.

The nature of traffic in the model is somewhere between TCP and UDP. It is not pure UDP due to the inclusion of ACKs and N-ACKs. At the same time, the traffic is not pure TCP because there is no notion of congestion control.

3 Simulation Results with varying source traffic distributions

The authors of [2] implemented a discrete event simulation of the model described in section 2. The simulation is written in Java using the JINQS API which is an extensible library for simulating multi-class queueing networks. A manual for the package can be found in [4].

3.1 Simulation Run Time

Before proceeding with the simulations, we first conducted an experiment to determine a suitable run time for our simulations. Since the truncated Pareto distribution, with its heavy tail, tends to reach the steady state fairly slowly as compared to the Poisson distribution, we simulated truncated Pareto distributed arrival process with different durations of simulation time. We found that after a lapse of 500 seconds since the beginning of the simulation, the MRT tended to stay constant for the next 500 seconds. So it was reasonable to assume that the system reaches steady state after 500 seconds. Consequently, for the rest of the paper (unless otherwise stated), the warm-up period has been considered as 500 seconds from the beginning of the simulation run. All measurements are reset at the end of the warm up period and the simulations are run for another 10 seconds over which period the statistics are extracted and used for further analysis. All simulations have been performed using the Camelot cluster [10] at Imperial College computing laboratory.

3.2 M/M/1 Queue with unit arrivals

We first simulated a simple M/M/1 queueing model with exponential arrival and service rates. The fixed parameters in the simulation were chosen as given in Table 3.1. These parameters are the same as used in the simulations in [2]. Packets arrive at the server as a Poisson process with mean arrival rate λ packets per unit time. By unit arrivals we mean that whenever there is an arrival, only one packet is injected into the system.

Table 3.1 Parameters chosen for Simulation of M/M/1 queues

λ	mean arrival rate	0.75×10^6 packets per second
μ_1, μ_2	service rate	10^6 packets per second
g_1, g_2	probability of corrupted data at A_1, A_2	0.02%
g_3	probability of corrupted data at receiver	0.1%
N_1, N_2, N_3	N-ACK delays from Router1 to sender, from Router2 to Router1, and from receiver to Router2	0.1 μ s
C_1, C_2	total buffer capacity of Router1, Router2	40
T_1, T_2, T_3	communication delays between sender and Router1, Router1 and Router2, Router2 and receiver	0.5 μ s
TimeOut	Sender time-out value	0.1 ms

Figures 3.1 and 3.2 show that with an increase in retransmission probability, the MRT also increases. The 95% confidence interval computed during 10 independent simulation runs is also shown.

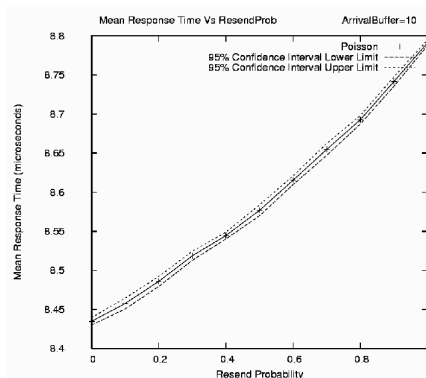


Figure 3.1 MRT versus retry probabilities for a tandem network with packet arrivals using $A_1 = 10$ and $R_1 = 30$

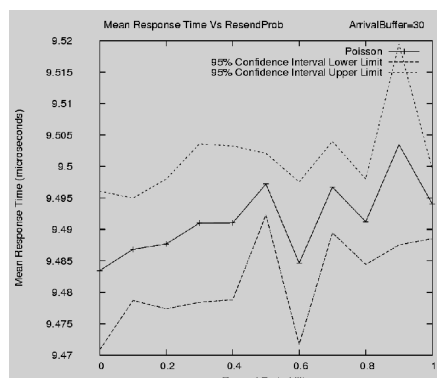


Fig 3.2 MRT versus retry probabilities for a tandem network with packet arrivals using $A_1 = 30$ and $R_1 = 10$

Figure 3.1 shows the results when using arrival buffers of size 10 and recovery buffers of size 30 whereas Figure 3.2 shows the results when arrival buffers are of size 30 and recovery buffers are of size 10. Resent packets that are eventually successful incur a higher transmission delay, and therefore result in raising the average of the response time. These two figures also depict that making arrival buffers larger results in a higher value of the MRT. This happens because the mean queue waiting time for the packets increases when the queueing buffer is large. However, an important point to note is that when arrival buffers are of size 30, the MRT is less dependent on the retransmission probability than when the arrival buffer size is 10.

This simulation was performed with the same parameters as mentioned in [2] but the results were found to be significantly different. While in our simulation, the MRT varies in the range $8.4\mu\text{s} - 8.8\mu\text{s}$ for the scenario in which arrival buffers are of size 10, the MRT in [2] is actually lower lying in the range $7.0\mu\text{s} - 7.5\mu\text{s}$ for the same set of resend probabilities. In the second case of A_i 30, the MRT in [2] was in the range $8.12\mu\text{s} - 8.16\mu\text{s}$ in [2] as opposed to being in the range $9.48\mu\text{s} - 9.505\mu\text{s}$ in our simulations. Further investigation revealed that the simulation in [2] was performed with communication and NACK delays of $0.05\mu\text{s}$ and $0.01\mu\text{s}$ respectively instead of the values stated in the paper.

3.3 M/M/1 Queue with batch arrivals

There are many situations where the approximating assumption of Poisson arrivals is reasonable: for example, a superposition of sparse, independent renewal processes is asymptotically Poisson and the net arriving traffic is indeed likely to come from many sources [2]. However, extensive studies of modern communication network traffic (including LANs and WANs) have revealed that modelling of network packet arrivals using traditional Poisson processes is not accurate. There is a high degree of correlation among packet arrivals which violates the Poisson assumption of independence. One major source of correlation is the prevalence of packets being sent in batches. We now assume that batches arrive at the server as a Poisson process with mean arrival rate λ **batches** (not packets) per unit time. We ran the simulation model with constant (deterministic) batch size distributions, keeping the batch size equal to 3. The rest of the parameters chosen for simulation are the same as in table 3.1. The arrival rate, however, is rescaled by a third so that the number of customers injected into the system is the same as in the case of unit arrivals.

Figure 3.3 shows a graph of the average MRT plotted against the retransmission probability with arrival buffer sizes equal to 10 and recovery buffer sizes equal to 30. Figure 3.4 shows the results when using arrival buffers of size 30 and recovery buffers of size 10.

Again we see that there is an improvement in the MRT when arrival buffer size is 10 and recovery buffer size is 30. This is due to a greater number of losses due to full arrival buffers when the arrival buffer size is small.

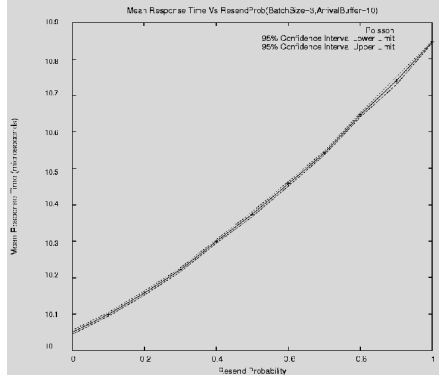


Fig 3.3 MRT versus retry probabilities for a tandem network with Poisson batch arrivals using $A_i = 10$ and $R_i = 30$

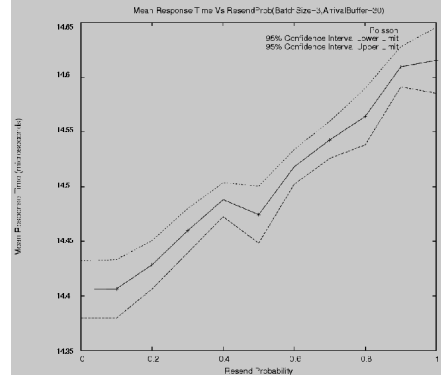


Fig 3.4 MRT versus retry probabilities for a tandem network with Poisson batch arrivals using $A_i = 30$ and $R_i = 10$

3.4 TP/M/1 Queue with unit arrivals

In order to study the effects of correlation and self-similarity on network statistics, we simulated truncated Pareto distributed packet arrivals keeping the service time distribution the same as before. The pdf of the truncated Pareto distribution is

$$f(x) = \frac{\alpha k^\alpha m^\alpha x^{-\alpha-1}}{m^\alpha - k^\alpha}$$

where α is the shape parameter, k is the lower bound and m is the upper bound of the truncated Pareto distribution, $\alpha > 0$, $k \leq x \leq m$, $0 < k < m$ and $k, m \in R$ and

The expected value of this distribution is

$$\mu = \frac{\alpha k^\alpha m^\alpha (m^{1-\alpha} - k^{1-\alpha})}{(1-\alpha)(m^\alpha - k^\alpha)}$$

As before, we first simulated unit arrivals for the Pareto distribution. For the purpose of comparison with experiments using Poisson arrivals in a later section, we matched the means of the two distributions and calculated the corresponding upper and lower bounds for truncated Pareto distribution. The simulation parameters are the same as given in table 3.1. The shape parameter α for the truncated Pareto distribution was chosen as 0.5, upper bound k was 1.333×10^{-8} seconds and lower bound m was 1.333×10^{-4} seconds.

Figures 3.5 and 3.6 show how the MRT varies with resend probability as arrival buffer sizes of both the routers are changed from 10 to 30. As observed in the case of Poisson arrivals, MRT is reduced when the arrival buffer size is small. This is because packets spend less time in the arrival queue.

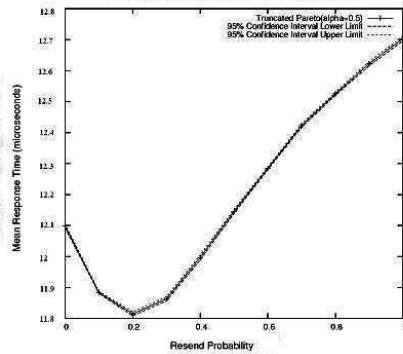


Fig 3.5 MRT versus retry probabilities for a tandem network with packet arrivals distributed according to a truncated Pareto distribution using $A_i = 10$ and $R_i = 30$

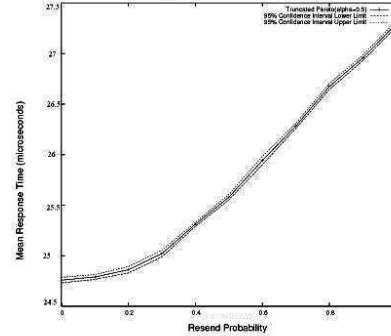


Fig 3.6 MRT versus retry probabilities for a tandem network with packet arrivals distributed according to a truncated Pareto distribution using $A_i = 30$ and $R_i = 10$

An interesting observation is that when $A_i=10$, we see a dip in MRT as resend probability increases from 0 to 0.1 and then a further dip from 0.1 to 0.2. It appears counter-intuitive at first. The model was re-run to check if the number of resent packets was actually decreasing with increasing resend probability but it was not found to be the case. It was verified that the number of resent packets does indeed go up with an increasing probability of retransmission. So, one reason for this peculiar behaviour could be that although the number of packets being resent increases, the packets that do get resent see fairly empty buffers (owing to the bursty nature of traffic) on arrival at the server and whiz through the system incurring lower delays. This results in bringing down the average of the response time of successful packets. When resend probability increases further, the system keeps getting more and more overcrowded and the MRT then continues to go up. This behaviour is not seen in the case of Poisson arrivals (with or without batches) because the traffic is less bursty and the likelihood that resent packets find empty buffers is very low. This finding, however, needs to be backed up with experimental results. One thing to do could be to re-run the simulation so that it records the MRT of the resent packets separately from the MRT of the packets that are successful in reaching the receiver in the first attempt. The results can then be analyzed to see if our hypothesis is correct or not. This unusual behaviour might also affect the optimization results presented in [2].

3.5 TP/M/1 Queue with batch arrivals

In order to observe the effect of batch arrivals in a heavy tailed distribution, we conducted experiments with truncated Pareto arrivals with a constant batch size distribution keeping the batch size fixed at 3. The parameters are the same as in section 3.5 with the exception of k which is chosen to be 4×10^{-8} packets per second and m which is 4×10^{-4} packets per second. This was done to keep the intensity of arrivals similar to what we had in the case of unit arrivals.

Figures 3.7 and 3.8 show how the MRT varies with resend probability as arrival buffer sizes of both routers are changed from 10 to 30. As observed in the case of Poisson

arrivals, MRT is reduced when the recovery buffer size is increased. This is owing to a decrease in the mean queue waiting time of the customers when arrival buffers are small.

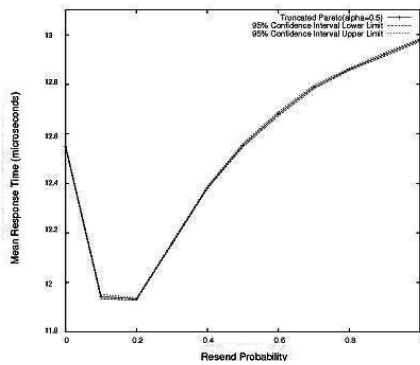


Fig 3.7 MRT versus retry probabilities for a tandem network with batch arrivals where batches are distributed according to a truncated Pareto distribution using arrival buffer size = 10 and recovery buffer size = 30

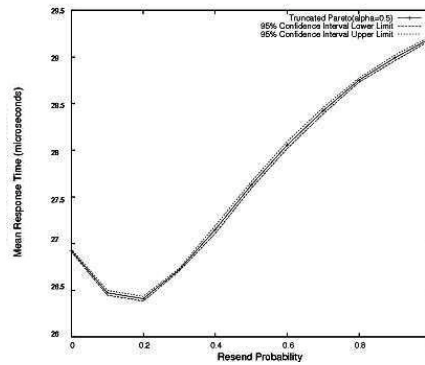


Fig 3.8 MRT versus retry probabilities for a tandem network with batch arrivals where batches are distributed according to a truncated Pareto distribution using arrival buffer size = 30 and recovery buffer size = 10

When we compare these results with those obtained in case of unit TP arrivals with arrival buffers of size 30, we see that for unit arrivals, the MRT showed an increasing trend but now it first decreases as the resend probability changes from 0 to 0.2, and then continues to rise. This could be because now the packets arrive in the form of batches of size 3 and whenever a batch witnesses a full arrival buffer, the whole batch of 3 packets is lost. Each of these packets is then sent for retransmission with the given probability. Thus the number of packets that are being resent is more than they were in the case of unit arrivals. Therefore these resent packets incur smaller delays leading to a lower average of the response time. Further increase in resend probability results in a more congested system thereby increasing the MRT of successful packets.

3.6 Analysis of receiver arrivals

In order to get an idea of what happens to the departure process that comes out of Router2, we analyzed the timestamps of the packets reaching the receiver for both Poisson and truncated Pareto distributed arrivals. The simulation parameters were the same as discussed for the case of unit arrivals. The receiver timestamps were collected during a simulation run of 15 minutes for each type of traffic distribution. Figure 3.9 shows a plot of the logarithm of the inter-arrival time distributions of the process at the receiver for Poisson arrivals on a semi-log plot i.e., plotted against time on a linear scale. The graph is almost a straight line which shows that the arrivals for the exponential external arrivals are still fairly exponential when they get to the last router. Figure 3.10 shows a similar plot for truncated Pareto external arrivals but it does not clearly tell anything about the distribution.

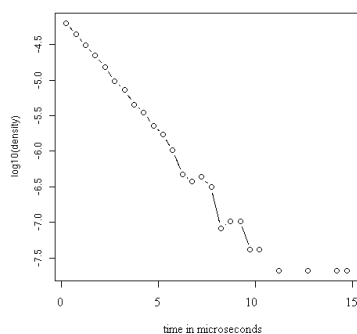


Fig 3.9 Logarithm of the density function of the inter arrival times of receiver arrivals when the input arrival process at the sender is exponential (with unit arrivals) plotted on a linear time scale

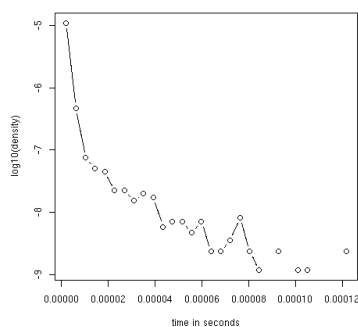


Fig 3.10 Logarithm of the density function of the inter arrival times of receiver arrivals when the input arrival process at the sender is truncated Pareto (with unit arrivals) plotted on a linear time scale

So, we analyze the graphs plotted in figures 3.11 and 3.12 which show the inter-arrival times of receiver arrivals for Poisson and truncated Pareto external arrivals respectively on a logarithmic time scale. For a Pareto distribution, the log-log plot should be a straight line. However, it is apparent from figure 3.12 that it is not a straight line which shows that it does not have a Pareto distribution at all. So, we conclude that while the exponential source arrivals maintain their exponential characteristics by the time they get to the receiver, the Pareto distributed external arrivals lose their power law characteristics.

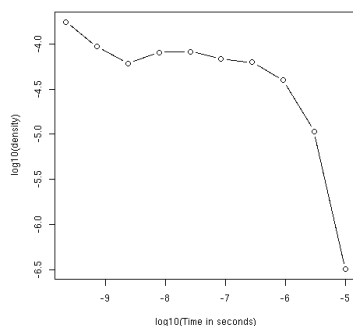


Fig 3.11 Logarithm of density function of inter-arrival times of receiver arrivals with exponential (unit arrivals) at the sender plotted on a logarithmic time scale

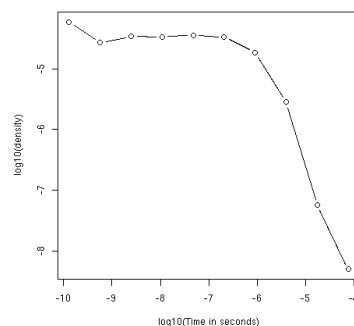


Fig 3.12 Logarithm of density function of inter-arrival times of receiver arrivals with truncated Pareto (unit arrivals) at the sender plotted on a logarithmic time scale

4 Poisson Arrivals with (truncated) Pareto Batches

Motivated by the studies in [3,7,11] that suggest a link between file/request size distribution and heavy tails, we made an attempt to simulate a similar situation in the context of our model. The model, parameters and simulation results follow in the subsequent sections.

4.1 Extended Model and Simulation Parameters

We consider an extension of the model in which requests of size P_T arrive at the sender and are forwarded to the first router. The sizes of requests (i.e., the number of arrivals in each request) are distributed according to a truncated Pareto distribution (and so the abbreviation P_T is used in figure 4.1). The time between the requests is exponentially distributed (as expressed by M for Markovian in figure 4.1). Both routers have exponential service times as before, and the rest of the network remains the same as discussed in Section 2.

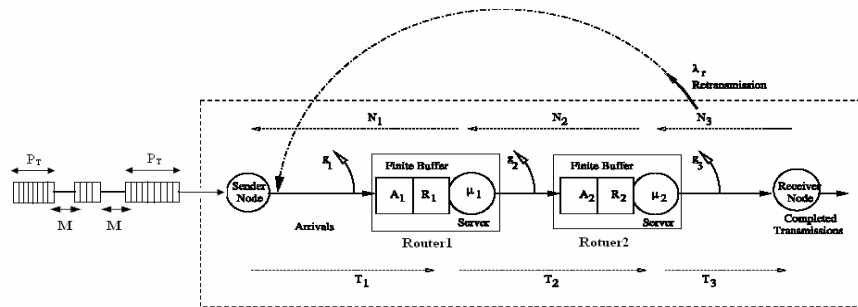


Fig 4.1 Model of a two node tandem network of mobile routers with Poisson arrivals that have truncated Pareto distributed batch sizes

The simulation parameters are the same as in table 3.1 except λ which is chosen to be 17050 arrivals per second. For the generation of truncated Pareto batches, the parameters used are as follows:

$$\alpha = 2.0, \quad k = 22 \text{ arrivals/second} \quad \text{and} \quad m = 220000 \text{ arrivals/second}$$

This results in a mean request size of 44 packets per second. As a result the mean arrival rate of the packets reaching the sender node becomes comparable to that used in the experiments carried out before. This has been done so that the results can be compared with those of the previous simulations. The motivation for having a mean request size of 44 packets per second comes from the discussion that follows.

A regular Ethernet frame uses a frame format that limits the size of the payload it sends to 1500 bytes. So any datagram bigger than 1500 bytes has to be fragmented. The maximum size permitted for an IPv4 datagram is 64 Kilobytes i.e., 65,536 bytes. A fragmentation of a datagram of that size to be sent over Ethernet would create roughly 44 packets.

The same simulation was first to be done with α chosen as 0.5 but that lead to an extremely low utilisation of routers even with very large buffers. Hence, we decided to use α as 2.0 for this new set of simulations, which gives a much higher utilisation percentage. For the sake of simplicity, the term Poisson-Pareto arrivals will be used to refer to the arrivals that are independent and identically distributed with exponential inter arrival times but have truncated Pareto distributed batch sizes.

4.2 Simulation results of Poisson-Pareto arrivals

Figure 4.2 shows a plot of the MRT as it varies with an increase in the resend probability when both arrival buffers are of size 10 and recovery buffers are of size 30. MRT first goes down as the resend probability increases from 0 to 0.1, and then starts going up. The same phenomenon was observed in case of truncated Pareto arrivals, and the same reasoning could be applied here. Due to bursty traffic, as the resend probability is increased from 0 to 0.1, more packets get resent but those that do get resent see fairly empty buffers and zip through the system incurring a smaller delay. Overall, this brings down the average response time because we add times smaller than the previous average. When the resend probability is increased further, the system keeps getting more and more over crowded thus resulting in a higher response time. The same reasoning can be applied to the results shown in figure 4.3 when the arrival buffer is changed to 30.

Comparing figures 4.2 and 4.3 we see that the MRT actually goes up when the arrival buffer is 30. This is because the mean queue wait increases as the arrival buffer can queue up more packets. Consequently, the average response time of the packets increases.

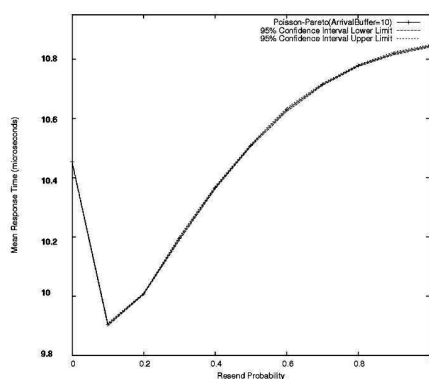


Fig 4.2 MRT versus resend probability for Poisson arrivals with truncated Pareto batches when the size of the arrival buffer is 10

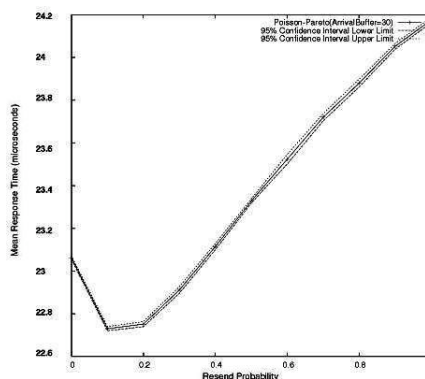


Fig 4.3 MRT versus resend probability for Poisson arrivals with truncated Pareto batches when the size of the arrival buffer is 30

4.3 Comparison of Poisson-Pareto with M/M/1 and TP/M/1 queues

In this section, we carry out a comprehensive comparison of the different types of arrival processes that we have discussed so far by comparing the MRT curves for each case. Figure 4.4 shows the MRT plot for $A_i = 10$ and $R_i = 30$. The MRT is lowest for pure Poisson arrivals and highest for Pareto batch arrivals. The Poisson-Pareto process has a MRT that is very close to the MRT of batch Poisson arrivals, whereas Pareto unit arrivals overestimate the response time. While the MRT of the Poisson-Pareto process is closely matched with that of Poisson batch arrivals when A_i is 10, the same is not true for the case of $A_i = 30$, see figure 4.5. The MRT in this scenario is much higher for Poisson-Pareto arrivals than for Poisson batch arrivals.

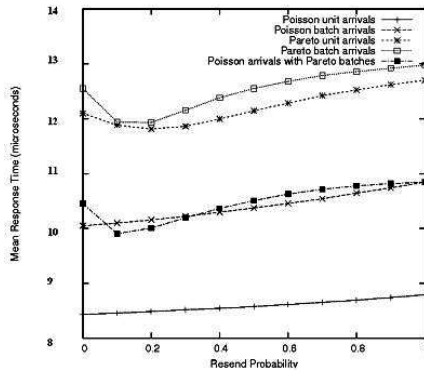


Fig 4.4 MRT versus resend probability for all types of arrivals with arrival buffer capacity equal to 10 packets and recovery buffer capacity equal to 30 packets

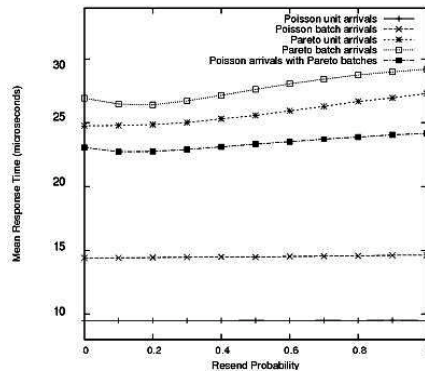


Fig 4.5 MRT versus resend probability for all types of arrivals with arrival buffer capacity equal to 30 packets and recovery buffer capacity equal to 10 packets

On further investigation, it was revealed that the reduced MRT in case of Poisson-Pareto arrivals is not because the process itself performs faster than the Pareto arrival process. The actual reason for a lower MRT of Poisson-Pareto arrivals is an increase in the loss rate and a decrease in server utilization as depicted by figures 4.6 and 4.7 respectively. Since most of the packets get dropped and only a very few of them are successful in reaching the receiver, the average of response times of these few packets is less as compared to the Pareto processes.

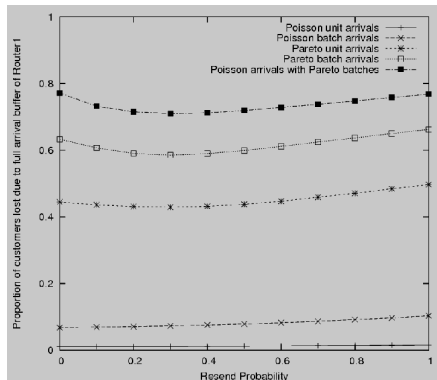


Fig 4.6 Proportion of customers lost due to full arrival buffers of Router1 versus resend probability when $A_1=10$ and $R_1=30$

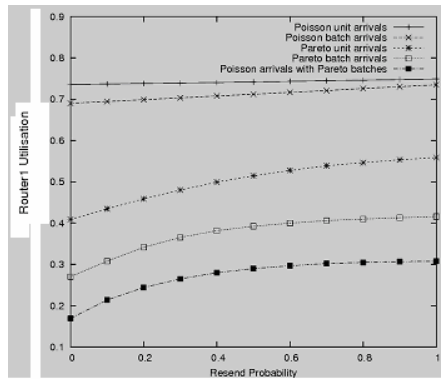


Fig 4.7 Utilisation of Router1 versus resend probability for arrival and recovery buffers of size 10 and 30 respectively

5 Conclusion

The simulations have been repeated for two different choices of arrival to recovery buffer ratio. When arrival buffers of routers are small, the MRTs are smaller than when the arrival buffers are large. The increased queueing delay due to large buffers increases the

MRT of successful packets. In case of large arrival buffers (of size 30), the results are less dependent on the resend probability than they are in case of small arrival buffers (of size 10), and losses are also reduced. However, the variance of response time (which was calculated and plotted for all these cases) for the arrival buffer size 30 scenario is much higher. Therefore, there is a trade-off between lowest possible packet loss and high variability of response time, and the choice of arrival to recovery buffer sizes will depend on the type of application and the parameter that we are trying to minimize.

While the pure Poisson assumption for the source arrivals makes the problem analytically tractable, it is not suitable to represent the highly correlated nature of modern internet traffic which is bursty over multiple timescales. Also, there is a high degree of correlation among packet arrivals which violates the Poisson assumption of independence. One major source of correlation is the prevalence of packets being sent in batches. So as the next step, we introduced the arrival of packets in the form of batches of fixed size while keeping the inter-arrival times exponential. The results of comparison with the case of unit arrivals depicted no significant change in the MRT.

In order to model the burstiness of arriving packets more accurately, a truncated Pareto distribution (with and without fixed size batches) was used to model the inter-arrival times of incoming packets. With this distribution, the undesirable effects (high mean and variance of response time, underutilisation of servers and high loss rate in particular) of heavy tails on performance measures become apparent. This is due to the bursty nature of arrivals in which there is little traffic over long periods combined with periods (bursts) of heavy traffic. When a burst of heavy traffic comes, the arrival buffers get full too quickly and the rest of the packets get dropped. Also, since the servers are idle for a longer time during the period in which the traffic is low, the server utilisation is reduced.

An analysis of the receiver arrivals shows that the external arrivals that are exponential maintain their exponential characteristics by the time they leave the queueing system. However, the truncated Pareto arrivals seem to lose their power law properties by the time they reach the receiver. This might have implications in the optimization framework and analytical solutions that were provided in [2] since the analysis was based on the assumption that the departure process from the first router is exponential. Our analysis shows that a similar assumption cannot be applied in the case of arrivals with truncated Pareto distributed inter arrival times.

Finally, an arrival process with exponentially distributed inter-arrival times and truncated Pareto distributed batch sizes was considered in order to represent the heavy tail properties of the file size distribution observed in real networks. The results show an appalling behaviour with packet losses being the maximum of all the discussed arrival types and the server utilisation being the minimum. The model fails to accommodate this type of arrival process and gives results that are not favourable in real networks.

6 Future work

The results presented in this paper are only based on simulation and there are no analytical results to validate them. So, one of the things that can be done is to validate the model using analytical tools.

The speculation that the MRT for Pareto arrival process (as well as for Pareto distributed batches) dips with increasing retransmission probability due to resent packets moving quickly through the network incurring lower delays, needs to be backed up with

evidence. This can be done, for instance, by calculating separately the MRT of the resent packets to see whether this actually is the case or not.

As explained before, pure TCP traffic has not been considered since there is no notion of congestion control introduced in the model system. So, another direction can be taken from here in order to study the effect of TCP congestion control on the model's structure and parameters.

The quality of random numbers generated by the Pseudo Random Number Generator (PRNG) is an important aspect in any computer simulation and depends on the period of the PRNG. The longer the period, the better the quality of the PRNG. The code used for the model presented in this work makes use of the `Math.random()` function of Java. `Math.random()` uses a 48-bit seed [12] which means that the period is very small. Another problem with using `Math.random()` is that it gives no control over the seed. So, a possible extension of this simulator would be to improve the PRNG used in the simulator. A very powerful PRNG algorithm is the Mersenne Twister [9] which is faster than `Math.random()`, has a period of $2^{19937}-1$ and gives a statistically more random output. Colt [13] contains the Java implementation of the Mersenne Twister algorithm. The simulator can be extended by including this library in the code and making it compatible with the existing program.

Acknowledgement

I would like to thank Dr. Uli Harder of Imperial College London for his continuous guidance and supervision during the course of this project.

References

- [1] Gulpinar, N., Harrison, P.G., Rustem, B., Pau, L.-F. An optimization model for a two-node router network. In: *Proceedings of the 12th international symposium on Modelling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004)*, Volendam, Netherlands. pp. 147–156
- [2] Gulpinar, N., Harder, U., Harrison, P.G., Field, A.J., Rustem, B. and Pau, L.-F. Mean-variance performance optimization of response time in a tandem router network with batch arrivals, *Cluster Computing*, vol. 10, pp 203–216, June 2007.
- [3] Field, A.J., Harder, Uli., Harrison, P.G., Network traffic behaviour in switched Ethernet systems. *Performance Evaluation* vol 58, pp. 243–260 November 2004
- [4] **Manual for JINQS API** <http://www.doc.ic.ac.uk/~ajf/Research/manual.pdf>
- [5] Leland, W., Taquq, M., Willinger, W., Wilson, D. On the Self-Similar Nature of Ethernet Traffic, *SIGCOMM*, 1993
- [6] Paxson, V., Floyd, S. Wide-Area Traffic, The Failure of Poisson Modeling, *IEEE/ACM TON*, 1995
- [7] Crovella, M., Bestavros, A. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes, In *Proceedings of SIGMETRICS '96: The ACM International Conference on Measurement and Modelling of Computer Systems*, Philadelphia, PA
- [8] Fowler, H., Leland, W. Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1139–1149, Sept. 1991.
- [9] **Mersenne Twister** <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>
- [10] **Camelot Cluster** <http://aesop.doc.ic.ac.uk/help/grail>
- [11] Irlam, G. Unix file size survey, <http://www.base.com/gordoni/ufs93.html>
- [12] **Java Random function** <http://java.sun.com/j2se/1.5.0/docs/api/java/util/Random.html>
- [13] **Java implementation of Mersenne Twister**
<http://dsd.lbl.gov/~hoschek/colt/api/ern/jet/random/engine/MersenneTwister.html>

On the componentization of queue solution methods

David J. Thornley*

Abstract

An important relationship between the matrix geometric and spectral expansion solution representations for calculation of the equilibrium state occupation probabilities of Markov modulated queues has recently been reported. The complementary strengths and weaknesses motivate translation between the two in a general analysis framework to combine their strengths. Further, coordinating the exactness of solution required in some circumstances with the low time complexity demanded by iterative optimization procedures motivates translation to and from approximate schemes. We illustrate the opportunity to translate into an approximate domain with an efficient statespace aggregation queue solution approach. We then draw a parallel between the extraction of the eigensystem from the state-aggregated output with performing an equivalent translation from the results of simulation. Translation to and from matrix exponential based queues, which provide an alternative to an explicit state space, is motivated by the opportunity for simplified traffic matching.

1 Introduction

The requirement for a formal and general treatment of the inter-operability of performability modeling approaches is clear. The fundamental theme of atomization, annotation and translation is well described by Deavours *et al* [7] in relation to the Möbius framework. This framework translates incoming model components to a common representation and solves these through simulation or numerical methods. Other unified modeling frameworks connect the components at other levels of abstraction, such as the TangramII integrated environment [6], but these still address similar basic requirements, such as impulse and rate reward modeling. The formulation of a model component, and its implementation for inclusion in such regimes requires the identification of the relationship between that models behaviour and the integration approach inherent to the framework.

Markov modulated queues are an important class of model commonly incorporated in such frameworks. Solution methods for queueing systems vary widely, but the matrix analytic and geometric approaches solve a large class of queues. Bini *et al*'s release of the SMC-solver [4] (written in Fortran 90) is an important contribution to the field of such solution methods for queues. The

*Department of Computing, Imperial College London, 180 Queen's Gate, South Kensington, London SW7 2RH, djt@doc.ic.ac.uk

algorithms provided [3] in the release of the tool gather results of the authors' research, and that of Neuts, Latouche and Ramaswami, and a contribution from Akar and Sohraby. Crucially it is also intended that further components be introduced by third parties. It provides a test bed for concrete comparison of methods, and a focus for discussion vital to efficient academic interaction and practical progress.

We are interested in issues surrounding the use of queue solution components in a network solution framework. Matrix analytic and geometric methods are applicable to a large class of queues described in recent work by Thornley and Zatschler [22]. This analysis defines bounds on the descriptions of queues for which a matrix generated solution is exact, and explains some circumstances in which numerical instability occurs. Note also that it clarifies exactly why spectral expansion is problematic around saturation through an analysis using generalized eigenvectors. This necessitates the use of alternative methods to the matrix analytic formulation under some circumstances which are defined as loci in the space of queue parameterizations. This in turn necessitates consideration of translation between methods to enable their use as substitutable components.

In this paper, we highlight shared structure in a range of queue solution methods, which motivates investigation of componentization of queue solution methods, for selection of the most effective .

2 Background

There are many practical methods in use for solving for the equilibrium state occupation probabilities of Markov modulated queues with or without batches (see [3] for examples). There are also a number of approaches to calculating sojourn or response times exactly, such as [11, 13] for the generalized batched queue [23], which construct the Laplace transform then invert this analytically), and approximate approaches to calculation of network sojourn times *e.g.* [10] which enable the calculation of network sojourn times. Each takes certain inputs, processes using a particular construction, and provides a specific output form. For example, each queue solution method will take as input a description of the arrival and service processes, and output terms which allow calculation of the steady state, either exactly or approximately, perhaps as moments of the distribution. Some important properties of the matrix geometric and analytic approaches have recently been reported [22] which clarify the relationships between spectral and matrix generated solutions. This enables us to extend the class of queues which are soluble exactly by analytic methods by the use of superposed geometrically batched arrival, service and removal (negative customer) processes in a single queue. This contradicts a long-held misapprehension that the matrix analytic approach was entirely general, as disproven in [22].

If we can find pairwise translations between values required for alternate methods, this will enable independent substitutions of these methods as components in a framework. As well as simplifying the construction of a network model, this will allow dynamic substitution of components, for example during iterative solution over a network, a stability analysis of such a solution, or an iterative optimization regime.

The matrix geometric representation writes a vector of state occupation probabilities for a given queue length as $v_j = vR^j$. The limitations of this,

proven in [22] can be illustrated succinctly by noting that R^j can be written as $W\Lambda^jW^T$ (where W and Λ are the standard matrix of eigenvectors and diagonal matrix of eigenvalue used when orthogonalizing a linear system), more clearly illustrating the basis for generating the queue solution. From spectral expansion of the queue, we see that there are up to NM eigenvalues, where N is the number of environmental phases, and M is the transition range in the most compact representation of the queue.

The three matrices commonly used for matrix geometric representation in matrix analytic methods, R , G and U are related to each other in a simple manner clearly explained in *e.g.* [3]. Thus, since queue solutions are generally iterative, these results may be passed to a new component brought in to improve efficiency, or to enable a richer description of traffic while maintaining throughput and the description of the processing. If we consider the relationship between a matrix geometric component's values and those of other methods we find interesting challenges, whether the methods be approximate such as Mitrani's recent introduction of the dominant eigenvalue approximation [16], or exact such as in the rapidly evolving use of matrix exponential distributions as exemplified by Akar's work (*e.g.* [1]).

It should be noted that the number of modulation phases need not be high to push the size of the eigensystem out of the range of matrix analytic approaches. A relatively simple processing node which takes a number of independent batched input streams, such as might be approximated using a maximum entropy solution for a generalized exponential description [14], requires a modified approach [22], since each independent batched process adds an eigenvalue.

3 QBD equivalent queues

Markov modulated queues soluble in tools such as the SMC solver exhibit a large region in which the transition pattern can be defined independently of that length. This results in a repeating pattern in the Chapman Kolmogorov balance equations:

$$\sum_{i=\max(0, j-f)}^{j-1} \mathbf{v}_i F(j-i) + \mathbf{v}_j [Q - \sum_{k=1}^{\min(f, j)} D(F_k) - \sum_{k=1}^{\min(b, L-j)} D(B_k)] + \sum_{i=j+1}^{\infty} \mathbf{v}_i M(I - \Phi) \Phi^{i-j-1} = \mathbf{0}$$

Vector \mathbf{v}_j holds the equilibrium state occupation probabilities of the queue at buffer occupancy j and modulation phase corresponding to the vector element. L is the maximum queue length, or waiting room size *including* the number of processors. We use a set of forward transition rates F_k for jumps of length $1 \leq k \leq f$, and a set of backward transition rates B_k . Each of L , f and b may be finite or infinite. Interestingly, f and b may be notionally infinite even if L is finite [23]. $D(A)$ is a diagonal matrix comprising the row sums of A . This notation is consistent with that used by Mitrani (*e.g.* [17, 16]). For more discussion of such Chapman Kolmogorov balance equations in the context of spectral analysis, see [16].

Thornley and Zatschler prove that for $f > 1$ or $b > 1$, the matrix generated solution may hold insufficient eigenvalues for a correct solution. However, it is noted that if the transition matrices take a geometric construction, *e.g.* $F_k = \Lambda(I - \Theta)\Theta^{k-1}$ and $B_k = M(I - \Phi)\Phi^{k-1}$, then the ensemble of balance equations for the queue can be manipulated using full rank operations to “eliminate” the infinite tails of the balance equation resulting in balance equations which arise from a localized equivalent transition structure. We therefore call a queue with structure amenable to this form of compaction a *QBD equivalent*. Thus, in balance equation below, Λ is a diagonal matrix of Poisson rates for an arrival process and Θ distributes the batch sizes. The service completions are in batches distributed according to Φ , arriving as Poisson processes in each modulation phase at rates given in M .

$$\sum_{i=0}^{j-1} \mathbf{v}_i \Lambda (I - \Theta) \Theta^{j-i-1} + \mathbf{v}_j [Q - \Lambda - M] + \sum_{i=j+1}^{\infty} \mathbf{v}_i M (I - \Phi) \Phi^{i-j-1} = \mathbf{0}$$

If we refer to the full equation above as f_j , then the QBD equivalent localized repeating equation is given by $c_j = (f_j - \Theta f_{j-1}) - \Phi(f_{j+1} - \Theta f_j)$. Replacing full equation f_j with compacted equation c_j in the ensemble preserves full rank (see [23] for further details). The resulting compacted equation for queue length j can then be written as follows:

$$\mathbf{v}_{j-1} Q_0 + \mathbf{v}_j Q_1 \mathbf{v}_j + 1 Q_2 = \mathbf{0}$$

This balance equation describes the compact equivalent of queues analytically soluble using a matrix basis. One of these queues is the geometrically batched BMAP/BMAP/1 queue described earlier. In its infinite form, it is soluble using matrix geometric approaches, and in its finite form, it can be analytically solved using an augmented matrix geometric form (see [22], or [2] for a derivation from a closely related point of view). This is because the ensemble of balance equations over the whole (possibly infinite) queue can be transformed using full-rank column operations (recall that the vectors are left-multiplied) on the ensemble of Chapman Kolmogorof probability flux equations for the whole queue, resulting in a compact – and most importantly *finite* – form which gives a quadratic matrix recurrence.

The manipulation regime described in [23] is a generalization of the approach above, and provides a compact equivalent of a queue with multiple superposed geometrically batched arrival, service and removal processes. In general the compact equivalent transition system covers a range of $M + 1$ queue levels, where M is equal to the sum of the largest equivalent forward batch transition range and the largest backward. This results in a linear homogeneous matrix recurrence relation of order M in square matrices of size N , so the characteristic equation has up to NM solutions. In the QBD equivalent, this is $2N$, N of which are inside the unit disc.

Thornley and Zatschler [22] explain how this leads to a system which will not collapse to a matrix quadratic, and discuss the resulting eigensystem size, proving that a matrix generated series – or pair of such series – is insufficient in the general case of such queues.

4 Componentization

The agility of deployment of the latest techniques in integrated environments is necessarily limited by the requirement for communication of the techniques, and the facility of adaptation of the interfaces to those techniques to the framework. We believe that this demands added emphasis in dissemination of novel performance results on the limitations as well as the power of the result. In addition, an explicit description of neighbourhoods of each method, in terms of which alternative methods we might translate to and from, will assist research at the level of integration frameworks. The neighbourhood of a technique is also defined in terms of the method to which we might translate, as it describes a region in the parameter space of the queue or network of queues being examined. In that region, it may be considered reasonable to use either method, and a decision to translate into one or the other will weight up the cost of translation against the computation cost in subsequent calculations. For example, if a queue will only stray into the neighbourhood of a competing solution method for a small number of computations, then the translation cost may outweigh the extra computation involved through not using the “better” method.

4.1 Translation

Many solution methods employ an iterative approach. During solution for perturbed network behaviour, perhaps as part of scenario exploration or parameter sweeps to test the stability of a system, intermediate results of those methods would ideally be stored for use in the next local iteration. If a component is to be substituted during this process, if the intermediate results can be passed on to the new component either directly or through translation, efficiency is improved. Use of alternate matrix analytic methods in this sense is ideal since the matrices G , R and U are interchangeable, and derive their distinct identities from the variant iterative methods.

In general, each translation may involve approximation and assumptions about the system. Information about these must be preserved in annotations. Complex solution systems will require meta-analyses of these attributes to track the error bounds of the solution methods, especially in non-feed-forward networks where the emergence of complex behaviour in the errors may be anticipated, which must be instrumented, as they may not be predictable. There may be identifiable patterns for establishment of a componentization which can be expected to behave well, and anti-patterns describing combinations and orderings which fail. This in turn offers an opportunity to construct a database, or ontology of suitable component combinations related to target applications.

4.2 Substitution

We believe that it is worth considering a scenario in which we formulate our queue solution methods such that they may be easily substituted dynamically in an overarching process of solution for network behaviour. For example, when a queue reaches saturation, we may choose to resign ourselves to a loss of detail for the sake of efficient feasibility of solution using an approximation.

For example, Mitrani [16] explains that the dominant eigenvalue method is particularly effective under heavy loading for an infinite queue. While it is clear

from the treatment that additional spectral components are important at lower loading, we might expect the use of a method based on a matrix geometric solution to be an effective approximator in two clearly identifiable scenarios. From [22] we note that for QBD equivalent processes, a matrix geometric solution is theoretically perfect, computationally efficient, and only slightly disadvantageous to numerical accuracy, when considering an infinite queue.

As well as translating between alternate exact methods, we may wish to move in and out of approximations. This is necessary for example when we wish to work from an initial estimate of network behaviour found from either simulation, or approximating analytic methods which provide either worst case analyses (*e.g.* [19]) or moment-based approximations (*e.g.* [14]). These are commonly necessary for a feasible solution which can include the effects of correlation in networks with feedback or overtaking [21].

We suggest that significant value will be added to results of novel or improved methods whose advantage is in high accuracy, low time complexity or helpful approximation in particular parameterization ranges, if we can identify complementary techniques in the literature which represent neighbours in the space of parameterizations of networks.

When approximate solution mechanisms are used, the precise character of the expected error must be maintained and passed on with the solution to subsequent processing elements to enable results derived from them to be presented with helpful error bounds. When asymmetric error bounds are guaranteed, for example if the solution is strictly pessimistic (*e.g.* [19]), then it may be necessary to include a periodic normalization or correction step in a global iteration step.

Matrix geometric and analytic solutions are exact for infinite quasi-birth-death queues and equivalents, and while the eigensystem of the single or double matrix solution is deficient for a more general class, it must be considered an important approximation: Mitrani demonstrates that a single eigenvalue is an operable approximation in heavily loaded queues, and the provision of a further $N - 1$ eigenvalues by the matrix G (which is N by N , the number of environmental phases), and in combination with matrix R , we have a total of $2N$ ([22] describes this combination in the solution of a finite QBD equivalent queue).

5 Example: ETAQA and spectral expansion

ETAQA [20] provides a particular solution form in which the first n queue lengths' vectors of state occupation probabilities are provided, and the sum of the remainder:

$$(\mathbf{v}_0, \dots, \mathbf{v}_{n-1}, \sum_{i=n}^{\infty} \mathbf{v}_i) \quad (1)$$

This represents an exact result, but is not a complete, detailed characterization of the equilibrium solution. Riska and Smirni have recently succeeded in generalizing the solution domain of ETAQA to include systems which allow batched downward transitions to more than one state and will output a solution for infinite queues of type GI/M/1 and M/G/1 and QBD. In their worked examples, generally $n = 1$.

We do not discuss the mechanism by which this is calculated except to note that ETAQA uses the same single matrix relationship between neighbouring

state occupation probability vectors as other matrix analytic approaches. This means that the solution is only exact for QBD equivalent systems. The solution structure however appears amenable to inclusion of more generating matrices, and output of state occupation probabilities at more levels of the queue.

We may also extract an approximation to the eigensystem, the simplest being a single exponential decay and eigenvector equivalent. Mitrani's approximation of the eigensystem using the dominant eigenvalue is driven by solution from the perspective of the system description, but ETAQA allows us to work from a solution for the steady state to find an effective mean (eigenvalue, vector) pair $(\bar{\zeta}, \bar{\psi})$ from the output $(\mathbf{v}_0, \dots, \mathbf{v}_{n-1}, \sum_{i=n}^{\infty} \mathbf{v}_i)$:

$$\epsilon^2 = \sum_{i=0}^{n-1} \left| \mathbf{v}_i - \beta \bar{\zeta}^i \bar{\psi} \right|_2 + \left| \mathbf{v}^* - \frac{\bar{\zeta}^n}{1 - \bar{\zeta}} \right|_2$$

Then set $\frac{\partial \epsilon^2}{\partial x} = 0, \forall x \in \{\beta, \bar{\zeta}, \bar{\psi}_1 \dots \bar{\psi}_N\}$

This is a basic least squares solution, whose effectiveness for a specific purpose would be optimized by synthesizing a suitable error weighting scheme in the normal manner. N is the number of modulation phases of the system, and $\bar{\psi}_i$ is the mean eigenvector's value in the corresponding phase i . In principle, we could include more approximating eigenvalues and vectors, but this would increase the size of the linear system we have to invert. In fact, it may be appropriate to use a single eigenvector with a number of eigenvalues, which would allow approximation of a heavy tailed distribution. The example G matrix entries shown in figure 5 of [4] indicate that the distribution among modulation states may be narrow overall, leaving the detail in the marginal queue length. Thus, thinking in terms of translation between methods has not only clarified relationships, it has also motivated analysis of a new, feasible approximation.

Riska and Smirni suggest that ETAQA does not allow the calculation of the exact steady state distribution of the queue, but with translation to an eigensystem representation, this can be readily mapped out for use in *e.g.* sojourn time calculation [11, 13] if sufficient queue levels are output. Thus translations may be valuable both for network equilibrium solution components, and for calculation of further measures.

6 Integration

It is tempting to suggest using the full eigenspectrum as the *lingua franca* in the community of queue solution methods, since it always expresses the complete behaviour. At the mathematical analysis stage, this and other formalisms stretching back to those used in [8] are helpful. However, it is well known that use of the explicit spectrum is problematic in a wide range of practical circumstances. In [22] we describe these circumstances, and outline some basic means by which they may be circumvented in the use of a matrix generated solution. This includes an analysis using generalized eigenvectors to allow proofs to include the locus of queue parameterizations which cause exact saturation. However, this does not help us in practice, since saturation is instantaneous with respect to changes in any given parameter. Instead, techniques from the numerical analysis literature must be introduced which are designed to cope

with vagaries of eigenspectrum distortion and instability. One important example of these is the use of the Schur form for matrices, which we see used in invariant subspace solution research. This may motivate the substitution of such an invariant subspace component during calculations near saturation.

Instability in the size of the eigenspectrum is effectively circumvented in the single eigenvalue approximation provided by Mitrani [16]. The efficacy of the approximation is analytically proven for heavy loading. However this may not be a suitable component for substitution during the progress of an iterative network calculation which is proceeding with a more expressive representation. Sudden loss of detail, especially if in a batched system, may cause shocks due to the strong effects of correlation in such circumstances [12]. This motivates the development of a formal approach to specifying the locus of queue parameterizations for which a given solution method is not just effective in itself, but crucially when it can act as a neighbour for other methods when substitution in a specified network solution regime is required.

Translations back and forth between exact forms and a variety of approximations may be warranted in an automated system. This may be used, for example, to seek out the range of network loads for which an existing deployment is effective, and then to experiment with feasible adjustments to expand that range. Commonly in an iterative scheme we establish an initial estimate, apply cheap updates until the solution is near, then use more sophisticated techniques to refine our view of the neighbourhood of the solution. This can be related to, for example, the Levenberg-Marquardt method for non-linear optimization, in which the computationally cheap approximation of steepest descent is used for an initial rapid approach to the solution, followed by phasing in a more expressive analytic approximation to the error surface to navigate the solution towards its exact goal.

6.1 Simulation

Simulation is an intuitive and successful approach to characterizing basics of network behaviour, including sojourn times. However, estimation of variance and higher moments of measures can be unstable. For this reason, analytic models are a commonly a necessary complement to simulation. If we consider the proposed theme of defining translations and annotations, we can compare the requirement to translate from simulation results to queue models with that from ETAQA. Queue length and system phase distributions can be noted from simulation, and in principle we can apply the same translation from this to an eigensystem representation.

If we wish to explore perturbation around a simulated behaviour, then we have to translate that simulated solution into a parameterization of a mathematical model (*cf.* [12]). The onus is then on the modeller to prove that the model is appropriate in a clearly defined manner, and to provide measures with distributions and/or bounds. Componentizing queue solution methods will enable this to be approached in a compositional manner, enabling the solution for these values using the mathematical annotations of the components, rather than having to synthesize a description from first principles for each network. The solution may still take the form of an iterative numerical solution, but as our understanding of the relationships between alternate methods improves, opportunities to find compact, analytical solution will arise, such as those lever-

aged by Schmitt *et al* [19] for an important class of approximations to provide pessimistic bounds.

6.2 Approximation

Mitrani [16] explores the use of a single, dominant eigenvalue/vector pair to solve a queue approximately. The effectiveness of the solution are clearly described. This is therefore directly amenable to rendering as a component with annotations describing the response of accuracy to load to guide a deployment mechanism in a larger performability measurement modelling task.

As mentioned above, we can solve for a “mean” eigenvalue from ETAQA output of state occupation probabilities which would tend to approximate the system with errors related to the approximation provided by Mitrani from an *ab initio* analysis of the queue description. We would be interested in this, for example, if we wish to calculate sojourn times for the queue. Mitrani’s approximation is sufficient at high utilization, in which the constant associated with the largest eigenvalue is dominant, and an ETAQA component may be substituted a lower loadings for an efficient solution mechanism which nevertheless allows us to capture the increased significance of the smaller eigenvalues by subsequent translation to a single mean eigenvalue/vector pair.

Matching of matrix exponential traffic and processing characteristics []. The number of eigenvalues required for the solution of a queue is generally NM , and a matrix geometric solution provides N of these (since $R = W\Lambda W^T$), so the solution may be approximate in a similar sense to that of Mitrani [16] but with a larger truncated eigensystem.

6.3 Matrix exponential queues

Fischer and Meier-Hellersten used matrix exponentials to analyse Markov modulated queues in an exposition of the range of Markov modulated queues [8]. More recently, Akar analyses queues explicitly formulated using matrix exponential distributions [1], and Fitzgerald *et al* [9] match matrix exponential random variables to real sequences. Akar also provides a state space method for solving queues with matrix exponential traffic [1]. These works, in combination with Akar *et al*’s work in invariant subspaces (we find [2] particularly informative), make clear the importance of matrix exponential approaches to provide feasible solutions for interesting queuing models, especially because this provides an additional key into matching real traffic essential to validating and applying theoretical work.

The matrix exponential distribution has a rational Laplace transform, rendering its analysis simpler in many cases. Also, its relationship to certain classes of state-space motivated queuing models is relatively straightforward. For example, the PH/PH/1 queue as analysed by Neuts [18], and later generalized by Latouche and Ramaswami [15] places a simple constraint on the parameters of the distribution, conveniently described by Akar [1]. Where the relationship between a state space model and the matrix exponential is clear, we expect the translation to either a spectral expansion or matrix geometric/analytic method to be straightforward.

7 Discussion and Conclusions

The establishment of a description for the locus of applicability, efficiency and accuracy of solution methods will assist in the identification of progress in the search for powerful solution mechanisms. Until we find the ultimate method which solves all queueing problems optimally, we must work to leverage the strengths of each individual approach. From the perspective of a team developing a given solution method, or teams contributing to approaches which occupy a similar interface (as intended for collaborative environments such as SMC-solver), a drive toward formalization of the relationship between that group of methods and others with different interfaces will add value, through the ability to hand over to another solution method in a manner provably appropriate over a well defined domain.

As Mitrani suggests, we must consider whether it is worth trying to solve analytic models exactly. The approximation he suggests enables approximate solution for a number of network measures. The feasibility of simulation as a proofing tool for system designs improves through academic community effort, and can rival analytic modeling in the tradeoff between accuracy and execution time. We note that the result of a simulation can resemble the output of ETAQA in terms of state occupation probabilities, for which we have suggested a translation to a spectral representation. This, in combination with an approximation to queueing rates will allow coordination of such results with analytic measures.

We have previously [22] looked at a partition of the space of Markov modulated queues which defines the range of applicability of a matrix generated solution. This range contains those queues whose system of balance equations may be compacted without approximation to give a matrix quadratic equivalent. An example of a queue which inhabits this range is the geometrically batched BMAP/BMAP, or MM CPP/GE/c/L form which corresponds to a queue initially examined in detail by Chakka and Harrison [5] without negative customers. This confirms a requirement for alternative solution methods for queues which fall outside this range. With the additional requirement for approximation in many circumstances, this motivates the establishment of translation mechanisms for analytic queue solution methods. Since the analytic common ground of the whole eigensystem has not been demonstrated generally efficient in practice, this requires pairwise analysis of methods which constitute “neighbours” in the space of queue parameterizations to find efficient translations. As new solution methods emerge, we suggest that value will be added, and future deployment made simpler, by specifying how their envelope of advantageous deployment interacts with their alternatives in both exact and approximate methodologies, and investigating how their inputs, intermediate results, and outputs may be interchanged.

References

- [1] N. Akar. Solving the me/me/1 queue with state-space methods and the matrix sign function. *Performance Evaluation*, 63:131–145, 2006.
- [2] N. Akar, N. C. Oguz, and K. Sohraby. Matrix geometric solutions of m/g/1 type markov chains: A unifying generalized state space approach. *IEEE Journal on Selected Areas in Communication*, 16, 1998.

- [3] D. Bini, B. Meini, S. Steff, and B. V. Houdt. Structured markov chains solver: algorithms. In *SMCtools '06 Pisa, Italy*, October 2006.
- [4] D. Bini, B. Meini, S. Steff, and B. V. Houdt. Structured markov chains solver: software tools. In *SMCtools '06 Pisa, Italy*, October 2006.
- [5] R. Chakka and P. Harrison. A Markov Modulated Multi-server Queue with Negative Customers - The MM CPP/GE/c/L G-Queue . *Acta Informatica*, 37:881–919, January 2001.
- [6] E. de S. Silva, R. M. M. Leão, and R. R. Muntz. Modeling, analysis, measurement and experimentation with the tangram-ii integrated environment. In *Valuetools '06 Pisa, Italy*, October 2006.
- [7] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Member, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The möbius framework and its implementation. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 28(10), October 2002.
- [8] W. Fischer and K. Meier-Hellstern. The markov-modulated poisson process (mmp) cookbook. *Performance Evaluation*, 18:149–171, 1992.
- [9] S. Fitzgerald, J. Placeb, and A. van de Liefvoort. Generating correlated matrix exponential random variables. *Advances in Engineering Software*, 37:75–84, 2006.
- [10] B. Gijzen, R. van der Mei, P. Engelberts, J. van den Berg, and K. van Wingerden. Sojourn time approximations in queueing networks with feedback. *Performance Evaluation*, 63:743–758, 2006.
- [11] P. Harrison. The mm cpp/ge/c g-queue: sojourn time distribution. *Queueing Systems*, January 2002.
- [12] P. Harrison, D. Thornley, and H. Zatschler. Geometrically batched networks. In *Proceedings of ISCIS 2002*, October 2002.
- [13] P. Harrison and H. Zatschler. Sojourn time distributions in modulated g-queues with batch processing. In *Proceedings of QEST 2004, 1st International Conference on Quantitative Evaluation of Systems*, pages 90–99, September 2004.
- [14] D. Kouvatsos and I. Awan. Entropy maximisation and open queueing networks with priorities and blocking. *Performance Evaluation*, 51:191–227, 2003.
- [15] G. Latouche and V. Ramaswami. The ph/ph/1 queue at epochs of queue size change. *Queueing Systems*, 25:97–114, 1997.
- [16] I. Mitrani. Approximate solutions for heavily loaded markov-modulated queues. *Performance Evaluation*, 62:117–131, 2005.
- [17] I. Mitrani and R. Chakka. Spectral expansion solution for a class of markov models: application and comparison with the matrix-geometric method. *Performance Evaluation*, 23:241–260, 1995.
- [18] M. Neuts. *Matrix Geometric Solutions in Stochastic Models*. Johns Hopkins University Press, Baltimore, 1981.
- [19] J. B. Schmitt and F. A. Zdarsky. The disco network calculator - a toolbox for worst case analysis. In *Valuetools '06 Pisa, Italy*, October 2006.
- [20] A. Stathopoulos, A. Riska, Z. Hua, and E. Smirni. Bridging ETAQA and Ramaswami's formula for the solution of M/G/1-type processes. *Performance Evaluation*, 62(1-4):331–348, October 2005.
- [21] D. Thornley and H. Zatschler. Analysis and enhancement of network solutions using geometrically batched traffic. In *Proceedings of UKPEW'03*, July 2003.
- [22] D. Thornley and H. Zatschler. Exploring accuracy and correctness in solution to matrix polynomial equations in queues. In *Proceedings of QEST 2006*, September 2006.
- [23] D. Thornley, H. Zatschler, and P. Harrison. An automated formulation of queues with multiple geometric batch processes. In *Proceedings of HETNETS'03*, July 2003.

Parallelization of a Simulation of a peer to peer market for Grid Computing

Fernando Martínez Ortuño*

Abstract

This paper focuses on a future Grid Computing market, which will be based on a completely distributed structure, covering all possible computing resources in the world. In this peer-to-peer market, nodes send messages to each other, offering or asking for the use of computing resources in exchange of real money. The paper proposes the parallelization of a previous simulation program that described this market, as well as the addition of new parameters and constraints.

1 Introduction

Grid Computing is a kind of distributed computing, formed by a network of heterogeneous geographically-distributed computing resources, that can be used to solve highly computationally-demanding tasks.

When tasks are submitted to the Grid in order to be solved, there must be a way of discovering which resources are available at that moment and which resources are allocated to solve each task. Since it is expected that lots of tasks are submitted at the same time, each of them requiring different kinds of resources and each of them demanding a different deadline, together with the fact that the Grid itself is expected to achieve a worldwide scope, the job of discovering and allocating the resources is not a trivial matter.

To solve this problem, several approaches have been taken; among them, the idea of applying economic theories [1].

Inside the economic solutions, some centralized auction systems have been used [2]. However, since these centralized systems have a single point of failure and prevent the system from becoming larger, they do not represent a long-term solution.

*Department of Computing, Imperial College London, fermaror@doc.ic.ac.uk

An opposite alternative is to establish a complete decentralized economic system: the Global Open Grid (GOG) [3], which consists of a world peer-to-peer network in which nodes trade for computing resources by sending and receiving messages to and from other nodes in the network. In GOG, the messages are originally sent to the node's neighbours and, if no matching offer is found, the messages are re-sent from the neighbours to their respective neighbours in turn, until a deal is found or until the time to live of messages expires. In this way, a node only has information about the messages he receives from other nodes and the deals he closes. Results of simulations of a simplified version of the GOG are presented in [4].

[4] showed that most of the parameters that were analyzed in the model reached an equilibrium independently of the network size. The evolution of prices, on the other hand, followed a permanent increase or a permanent decrease, due to the lack of constraints in the nodes' budgets.

This paper is built upon the simulation model used in [4], and it presents some new additions and modifications, as well as a possible first parallel version of the model.

2 MaGoG

This work assumes the MaGoG system as the future basis for the Grid Market. MaGoG stands for 'Middleware for Activating the Global Open Grid', and its complete description can be found in [3]. Basically, MaGoG is a peer-to-peer based architecture that considers all computational resources in the world (from mainframes to mobile phones) as part of the Grid Market. The architecture is based on three main concepts: Catallaxy, the 'small-world' networks and double message flooding.

The Catallaxy paradigm of Austrian School economist Friedrich von Hayek [5, 6] states that, in a decentralised competitive market where all entities aim to maximize their utility, a spontaneous 'order for free' will emerge via a complex adaptive process, creating a stable state in the system without the need of a central omniscient auctioneer. We do not expect, however, that this stability in the system implies a stability in prices, since prices in the Grid Market will permanently fluctuate in an endless game according to variations in offer and demand, as it takes place in any other market, such as Currency Markets, Stock Markets and Futures Markets. Consequently, the system will try and look for the equilibrium state according to the market situation in a particular instant of time, but this situation may change in the next instant of time, and so the system will change its direction in an endless game in order to find the new equilibrium state.

The second concept MaGoG is based on is the idea of the 'small-world' networks, which is actually based on an observation by Stanley Milgram. According to this observation, there are, on average, only six degrees of separation between

any two individuals on Earth. Therefore, even in a potential future situation where all computing resources in the world are connected in the Global Open Grid, it will be possible for any node of the Grid Market, by passing messages between peers, to reach any other node in the Grid with an average of only six hops.

The third key concept MaGoG is based on consists of double message flooding, together with the possibility of closing deals between two parties in a third party node. This means that, in MaGoG, both buyers and sellers send their messages to their peers in order to find a deal and, if in this process two matching offers meet in a third party peer, a deal is closed between them in that third party peer, avoiding further propagation of the first two nodes' offers in the Grid. In other words, every single peer of the Grid provides a pub, or trading floor, where messages (offers/orders) that were originally issued from other peers in the Grid can meet and close deals. Consequently, this increases the efficiency of the system and reduces considerably the volume of messages within the network.

3 Our simulation model for MaGoG

The model we present in this paper is based upon the one described in [4], which simulates the MaGoG system. However, from a programming point of view, the model we introduce in this paper presents two basic differences with the one in [4], specifically: the re-writing of the simulation program with an object-oriented approach on the one hand, and its parallelization on the other hand.

An object-oriented approach is considered more suitable, since nodes can be represented by objects, and it gets closer to an agent-based simulation. Furthermore, the object-oriented language facilitates the implementation of new features, making the model more versatile.

The parallelization of the program will allow us to perform a simulation with a larger number of nodes, since it will be possible to increase the memory the program uses by making use of several processors' memories. The fact of making a simulation with a larger number of nodes will permit us to get results that are nearer the reality, where a global network connecting all possible computing resources in the world is expected to emerge [3].

On the other hand, the system to make deals between the messages that, at a particular moment in time, are present in a node, is changed (compared with [4]). Actually, since the messages a node's buffer receives come from nodes who do not know, at least in principle, the prices the other nodes are asking/bidding, this situation resembles the pre-trading status that takes place some minutes before the stock exchange opens, when the opening price is decided in an initial auction, and the order book is invisible for the traders. Deals in nodes' buffers are made now in this way, so that nodes' buffers become real trading floors.

A significative addition to the model presented in [4] is the establishment of a system with three different types of agents: a group of permanent buyers, a group of permanent sellers and a group of traders. The buyers model the institutions that will demand computing power for their simulations, and they will always want to buy resources; however, their buying attitude may change depending on the current market price and their own budgets. The sellers model all those people who will connect their home resources to the Grid, in order to hire them when they do not need to use them (for instance at night); they have no intention of buying computing resources and they just want to get some money by hiring their computers. Finally, the traders are able to send buying and selling orders to the Grid; their objective is to make money from the Grid market by gaining the difference in price with buy/sell operations. We expect that this system will help in bringing a more rational price evolution.

Although we do not present simulation results in this paper, we plan to obtain them soon and publish them in a future paper.

3.1 The object oriented approach

The model we present in this paper has an object-oriented approach, in contrast with the structural programming model that was presented in [4]. In this model, we define the class Node, which includes among its members the attributes that are common to all nodes in the network, such as a unique identifier, a list of the node's neighbours, the price the node is asking/bidding, etc. The class Node also includes a long list of methods that can be invoked by the Node; these methods are used to deal with the messages in the trading floor (pub/buffer) of the node, to deal with the status of the node, to forward messages to the node's neighbours or to make the auction among the messages in the node's floor and to match deals between them.

From the base class Node, we construct three other derived classes: the class Buyer, the class Seller and the class Trader. Objects of the class Buyer can only issue buying messages, objects of the class Seller can only issue selling messages, and objects of the class Trader can issue messages of both types. These three derived classes inherit the members of the class Node and they also add new methods that are exclusive of their classes. For instance, the class Trader adds some trading strategies.

3.2 The way of making deals

The way of making deals that is implemented in this version of the model differs from the one used in [4]. In [4], among the existing messages inside a node's trading floor, any two messages from a different type of node (buyer and seller) that accomplish the price requirement (the price of the buyer is equal or higher than the price of the seller) come to an agreement at a price which is the average between the price the seller is asking and the price the buyer is bidding. In the

present model, the matching of orders is made by an auction among all buying and selling messages whose prices overlap. The result of the auction determines the auction price and the consequent matching between the messages in the trading floor; all deals in this case are made at the same price: the auction price.

The algorithm we use to make the auction in the trading floor is similar to the one used to calculate the opening price at the stock exchange. In particular, we follow the algorithm used by the Australian Securities Exchange [7]. This algorithm applies four conditional principles in order to decide the auction price, which means that, if Principle 1 gives a clear result, the algorithm stops there, whereas if no clear result is achieved with the first principle, then Principle 2 is applied and so on. An explanation of the algorithm can be found in [7]. Schematically, the four principles to determine the auction price are:

- Principle 1: Determining the Maximum Executable Volume
- Principle 2: Establishing the Minimum Surplus
- Principle 3: Ascertaining where the Market Pressure exists
- Principle 4: Consulting the Reference Price

4 The parallel version

In this paper we aim to introduce a first parallel version of the model presented in [4]. Although this first parallel version is still in progress, we present in this section some of the ideas that may be applied.

The objective of making a parallel version has to do with the fact that, by using memory from several processors, we can increase the global memory of the system and, consequently, make simulations with networks of a larger number of nodes. Therefore, the first attempt is to make a parallel version by using distributed-memory multiprocessors. In particular, we will use MPI (Message Passing Interface) to make communication between processors possible.

The problem when trying to make a parallel version of this system is that, in our case, we are not dealing with the classical structure that can be easily parallelized. In other words, this is not the case where a master process, that is in charge of the whole system, divides a heavy task into smaller ones and sends each of these tasks to slave processes that solve them without having to communicate in excess with each other. When each of the slave processes finishes its task, they send the results back to the master process, that gathers all the information and is able to solve the initial problem. In our case, in contrast, there is no any big task to solve, and we have many nodes that communicate frequently with each other, so communication between processes will also be frequent.

The idea we propose in this paper in order to parallelize the system is to divide the network that forms the Grid into several parts, each part containing a group of nodes. Each of those parts will be handled by a different processor, that will be in charge of updating the nodes that belong to that part of the network. As long as nodes in one part of the network do not need to send or receive data from nodes that belong to a different part of the network, communication between processors will not be necessary; the contrary otherwise.

There are two occasions where nodes have to exchange information between them. One of the occasions occurs when a node has to forward the messages in its floor or send its own messages to its nearest neighbours. The other occasion takes place when, at the time of closing deals, it is necessary to know whether the owners of the messages that are going to close a deal are still in the unsatisfied state or not. Consequently, these will be the two occasions when processors may need to communicate between each other.

Since MPI allows to define new data types that can be used as the basic unit of communication between processors, we will create two new data types, that correspond to the two occasions when processors need to exchange information between them. One new type of data will correspond to the case when a processor needs to ask another processor what the state of a node is. For this first case, the type of data will be formed by two fields: one field that identifies the node and another field that informs about the state of that node. The other type of data corresponds to the case when processors send a message issued by a node to another processor. For this second case, we create a new type of data that has the same structure as the message that nodes send between each other. Whenever the processors need to communicate with each other, they will use one of these structures in order to send and receive information.

4.1 A practical example

To illustrate how the parallel version may work, let us consider a network of 32 nodes, which are handled by two processors: processor 0 and processor 1. In order to facilitate the explanations, we will call processor 0 *he*, whereas processor 1 will be called *she*.

As explained above, the network is divided in several parts. In this case, as there are only two processors, we divide the network in two parts, so that half the nodes in the network will be controlled by processor 0 and the other half will be controlled by processor 1 (see example in Fig. 1). Different partitioning algorithms may be used to decide which nodes must belong to each part of the network; we will not consider that discussion here.

Once the network is divided, the two processors start executing the code concurrently. Consequently, each of the processors picks up a node that belongs to their part of the network. The two processors carry out all the updates and tasks in that node than can be done independently of the behaviour of the other processor. And it is just before making the auction with the messages in the

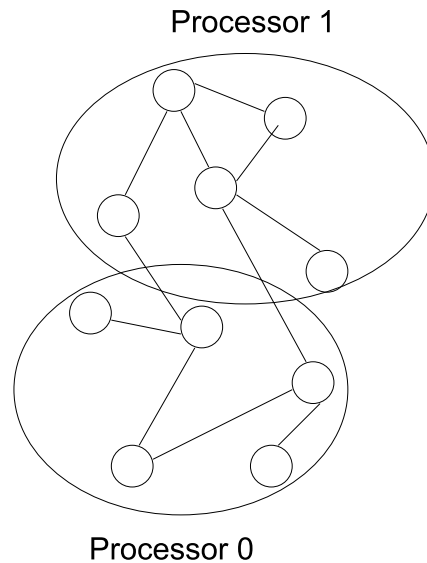


Figure 1: The network is divided in two parts. Each of the parts is controlled by a different processor.

node's trading floor when a first synchronization between the two processors is necessary. Making the auction and assigning deals between messages in the node's floor is a critical part that can't be executed simultaneously by two processors, since the two nodes that are being updated could have messages from the same node in their floors, which could lead to the same node closing the same deal with two different nodes at the same time. To avoid this from happening, we do not allow the two processors to make the auction in their respective nodes at the same time. In order to achieve this objective, we synchronize both processors by making use of the blocking mechanism inherent in some functions of MPI. Figure 2 shows how the two processors synchronize and communicate with each other in every lap of the loop.

We assume, for simplicity, that processor 0 has preference over processor 1 when it comes to make the auction, so processor 0 will make the auction of the messages in its node's floor before. Following the diagram of Fig. 2, processor 0 starts the auction process. In this version of the model, before making the auction, the processor checks, first of all, that the messages that are going to participate in the auction do not have their TTL expired, on the one hand, and that the messages do not belong to nodes that are in the satisfied state at this moment, on the other hand. The TTL of the messages is directly checked, whereas the state of the nodes that have issued the messages requires communication with those nodes.

Therefore processor 0, after deleting old messages, checks who the owners of the remaining messages are. If the owners are nodes that belong to the same part of the network processor 0 is analyzing, he checks their state immediately.

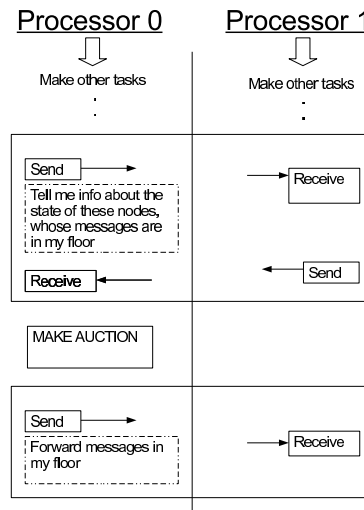


Figure 2: Schematic view of the synchronization and communication between the two processors by using the blocking mechanism of the MPI functions in every lap of the loop.

If the owners are nodes that belong to the other part of the network, processor 0 makes a list of these nodes, and sends a single request to processor 1, asking her for the state of those nodes. If no messages from nodes in the other part of the network are present in the trading floor that is being analyzed by processor 0, processor 0 sends an empty request to processor 1. This is due to the fact that, on the other hand, processor 1 has been blocked before making the auction of the messages on her floor by a call to a receive function. Consequently, she is expecting a request from processor 0 in order to continue with the execution of the code. This is the reason why processor 0 will send, in any case, a request to processor 1. After the send call is completed by processor 0, he continues executing code and is blocked by a receive function, that forces him to wait for the information that processor 1 has to send him.

On the other hand, processor 1 will have picked up a node in her part of the network, she will have updated all the parameters that do not require communication with processor 0, and she will have been blocked by a receive call, that forces her to wait for the request from processor 0. Immediately after she receives this request, processor 1 checks the state of the nodes in her part of the network processor 0 has asked her to check, she packages all the information and sends a unique send call to processor 0 with that information.

Processor 0 is then unblocked after receiving the send call from processor 1, and now he has all the necessary information to make the auction with the messages in the node's floor he's analysing. After making the auction, some of the messages in the node's floor will not have made any deal, so they are forwarded (if they have not been forwarded from this node before) to the nearest node's neighbours. Some of these neighbours may belong to the other part of

the network, so process 0 makes a package with all messages that need to be sent to nodes in the other part of the network, and sends the whole package to process 1. If no messages have to be sent to the other part of the network, process 0 creates an empty package and sends it to processor 1. In this case, the basic unit of information that is sent between processors is the data type that was defined for the messages that nodes send to each other.

On the other hand, processor 1, who was blocked by a call to a receive function, receives the information package from processor 0, she unpackages it and sends each of the messages to the nodes in her part of the network the messages are addressed to.

At this moment, processor 0 has successfully completed the auction with the messages in the node's floor he was analyzing, so now processor 1 is the one that can enter this critical part of the code and make her auction. For the case of processor 1 making the auction, the above actions are repeated exactly in the same way, but inverting roles between the two processors.

This naive mechanism of synchronization guarantees the simultaneous update of the two nodes by the two processors without errors in the trades. The mechanism can be extended to the use of more processors by the partitioning of the network in more parts, each of them controlled by one processor.

5 Conclusion

This paper has proposed a new approach to the simulation model presented in [4]. The re-writing of the model with object-oriented programming increases its flexibility and allows us to incorporate new parameters easily, in particular in relation to the attributes of the nodes.

The different way of making deals at the pubs of the nodes makes the trades fairer. Although we do not present simulation results in this paper, preliminary results show that the price evolution might follow, for this new version of the model, a tendency that is independent on the kind of network, in contrast with what happened in [4]. The way of making deals is, consequently, decisive in the price evolution of the system.

On the other hand, we have described a possible first parallel version of the system. Despite the difficulties associated with a model in which communication between processes is continuous, we have shown a possible way to avoid conflicts while processors try to execute simultaneous trades.

Furthermore, the parallel version of the system will allow us to study the evolution of the system locally, i.e., the different evolution of the system depending on the area of the network each processor is analysing.

We expect to obtain results of this version soon.

Acknowledgments

I would like to thank Nicholas Dingle and Uli Harder for ideas about the parallel version and the igrph package.

References

- [1] I. E. Sutherland. A futures market in computer time. *Commun. ACM*, 11(6):449–451, 1968.
- [2] Uli Harder, P.G. Harrison, Maya Paczuski, and Tejas Shah. A dynamic model of a GRID market. In *Proceedings of Invited poster at ACM/IEEE Mascots 2004 - ISBN 0-7695-2251-3*, October 2004.
- [3] Colin Richardson. Growing the Global Open Grid: Design Brief and Middleware Architecture. Technical report, The Internet Centre, Imperial College London, 2007.
- [4] Uli Harder and Fernando Martínez Ortuño. Simulation of a peer to peer market for grid computing. In *The 15th International Conference on ANALYTICAL and STOCHASTIC MODELLING TECHNIQUES and APPLICATIONS, ASMTA 2008*, volume 5055 of *Lecture Notes in Computer Science*, pages 234–248, 2008.
- [5] Oscar Ardaiz, Pau Artigas, Torsten Eymann, Felix Freitag, Leandro Navarro, and Michael Reinicke. The catallaxy approach for decentralized economic-based allocation in grid resource and service markets. *Applied Intelligence*, 25(2):131–145, 2006.
- [6] Torsten Eymann, Boris Padovan, and Detlef Schoder. The catallaxy as a new paradigm for the design of information systems. In *Proceedings of the 16th IFIP World Computer Congress, Conference on Intelligent Information Processing*, 2000.
- [7] Carole Comerton-Fordea and James Rydge. The influence of call auction algorithm rules on market efficiency.

State-Space Size Estimation By Least-Squares Fitting

Nicholas J. Dingle William J. Knottenbelt *

Abstract

We present a method for estimating the number of states in the continuous time Markov chains (CTMCs) underlying high-level models using least-squares fitting. Our work improves on existing techniques by producing a numerical estimate of the number of states rather than classifying the state space into one of three types. We demonstrate the practicality and accuracy of our approach on a number of CTMCs generated from three Generalised Stochastic Petri Net (GSPN) models with up to 11 million states.

1 Introduction

A vital component of many correctness and performance analysis techniques is the explicit enumeration of the state space underlying a high-level model such as a Petri net or a process algebra specification. An early indication of likely state space size is beneficial in a number of ways. For example, it provides the user with a good idea of the computational resources (CPU time, number of CPUs, memory, disk space etc.) that will be required to complete the analysis process. If the estimate suggests that currently allocated resources are inadequate, the process can be restarted early with increased resources. The latter is particularly useful in utility computing environments where charges are levied on a CPU hour basis. Alternatively, a large estimate may persuade the modeller to apply an alternative exploration or analysis strategy (e.g. using probabilistic methods [9, 11], or “on-the-fly” [2] approaches) or to revisit the level of abstraction employed in the model.

However, with the exception of specialised models with restricted structure [12], there are few known techniques for estimating state space sizes from high level models in the literature. An important recent work, and the inspiration for our present investigation, is the paper of Pelánek and Šimeček [10] in which various methods for estimating state-space sizes are discussed. The methods fall into two main categories: those based on state sampling and those based on the attributes of breadth-first state-space exploration.

The sampling-based approach works by taking two samples of the state space, each containing s states, and comparing the states in each. Some number, x , will appear in both and the ratio x/s is then used to classify the size of

*Department of Computing, Imperial College London, 180 Queen’s Gate, London SW7 2BZ, United Kingdom. Email: {njd200,wjk}@doc.ic.ac.uk

```

begin
   $A = \emptyset$ 
   $E = s_0$ 
   $F.insert(s_0)$ 
  while  $F(\text{not empty})$  do begin
     $F.remove(s)$ 
    foreach  $s' \in \text{succ}(s)$  do begin
      if  $s' \notin E$  do begin
         $F.insert(s')$ 
         $E = E \cup s'$ 
      end
       $A = A \cup \text{id}(s) \rightarrow \text{id}(s')$ 
    end
  end
end

```

Figure 1: Breadth-first search algorithm for state-space exploration [8].

the total state space into one of three categories. The samples can be generated by breadth-first search, depth-first search or a random walk.

The estimation from the attributes of breadth-first search also aims to classify the overall state-space size into the same three categories based the number of states in the first k levels of the search. Estimation by human judgment, by classification trees and by neural networks was conducted, as well as an investigation into combining these classification methods with the results from the random-sampling approach to improve accuracy.

The major limitation of [10] is that the authors explicitly avoid estimating the total number of states and instead confine themselves to placing the estimated total state-space size into one of three categories (i.e. those models which can be handled easily, those which may require state-space reduction or parallel generation and those which are too large). In contrast, this paper presents a method for dynamically estimating the number of states in the underlying state-space of a high-level model. Our method uses least-squares fitting from the number of states currently observed during the breadth-first state generation process. We demonstrate the accuracy of this technique on a number of state spaces generated from three high-level Generalised Stochastic Petri Net (GSPN) models.

The remainder of this paper is organised as follows: Section 2 briefly presents the breadth-first state-space generation algorithm used the DNAmaca [7] steady-state analysis tool, before Section 3 describes the method of least-squares fitting. Section 4 then introduces the three GSPN models considered and demonstrates how the fitting method can be used to predict accurately the total number of reachable states whilst the state-generation process is underway. Finally, Section 5 concludes and suggests directions for future work.

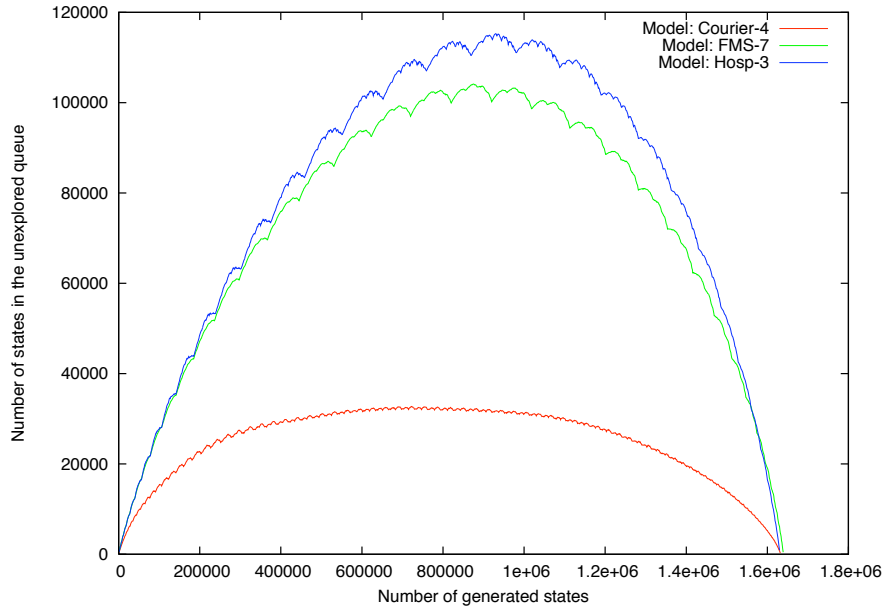


Figure 2: The number of unexplored states during the state-space generation process for three similarly-sized GSPN models.

2 Breadth-First Search

In DNAmaca [7], the breadth-first search algorithm shown in Fig. 1 is employed to explore the model's state-space. This starts from an initial state s_0 and uses a FIFO queue (F) and a list of explored states (E) to generate the state graph (A). The functions `insert()` and `remove()` add a state to and extract a state from F respectively, while `succ(s)` returns the set of successor states of s . The function `id(s)` returns a unique sequence number for state s . DNAmaca uses a probabilistic hash-based scheme [7, 8, 9] to store E in a memory-efficient and easily-searched manner. Breadth-first search is favoured over depth-first search as it allows the continuous time Markov chain's (CTMC's) generator matrix Q to be created and written to disk row-by-row without the need to maintain more than one row in memory.

A guide to the progress of the state-space generation process can be gained during the execution of the BFS algorithm by examining the number of states in F . Fig. 2 shows how the length of F changes over the course of state-space generation for three of the models described in Section 4.

3 Least-Squares Fitting Method

We observe that the shape of the graph in Fig. 2 can be approximated by a curve with equation $y = ax^2 + bx$, for some values of a and b . In order to find these values we use the GNU Scientific Library (GSL) [4] to perform multiparameter fitting using least squares. This fits a model of p parameters to n observations – in our case, there are two parameters (a and b) and the observations are the

Model Name	k	Tangible States
<i>courier2</i>	2	84 600
<i>courier3</i>	3	419 400
<i>courier4</i>	4	1 632 600
<i>courier5</i>	5	5 358 600
<i>fms5</i>	5	152 712
<i>fms6</i>	6	537 768
<i>fms7</i>	7	1 639 440
<i>fms8</i>	8	4 459 455
<i>fms9</i>	9	11 058 190

Table 1: Number of tangible states in the Courier and FMS models in terms of the sliding window size/ the number of unprocessed parts (k).

Model Name	Patients (P)	Nurses (N)	Doctors (D)	Ambulances (A)	Tangible States
<i>hosp1</i>	7	2	2	1	54 228
<i>hosp2</i>	10	2	2	1	561 704
<i>hosp3</i>	11	4	2	2	1 630 905
<i>hosp4</i>	13	4	2	2	5 728 971

Table 2: Number of tangible states in the hospital model in terms of the number of patients (P), nurses (N), doctors (D) and ambulances (A).

current number of unexplored states in F during the BFS process. As the total number of states can be very large, if we were to take observations at each iteration of the BFS process we would potentially have several million and so, to keep the problem size small, we only observe the number of unexplored states once for every 10 000 states generated.

The first step in the fitting process is to express the problem in matrix-vector form $y = Xc$ where y is the vector of n observations, X is an n -by- p matrix of the predictor variables and c is the vector of p unknown best-fit parameters. As we are fitting a polynomial of degree 2 we define $X_{ij} = x_{ij}^2$ for $0 \leq i \leq (n-1)$ and $0 \leq j \leq (p-1)$. We then employ the `gsl_multifit_linear()` routine (which implements the modified Golub-Reinsch singular valued decomposition algorithm [5] with column scaling) to find the values of a and b which yield the best fit to the observations.

4 Results

To demonstrate the power of our approach, we present results using three Generalised Stochastic Petri Net (GSPN) models. GSPNs are attractive as they typically feature a small number of parameters (tokens) which can be easily varied to produce CTMCs of differing sizes. The GSPN in Fig. 3 models the ISO Application, Session and Transport layers of the Courier sliding-window

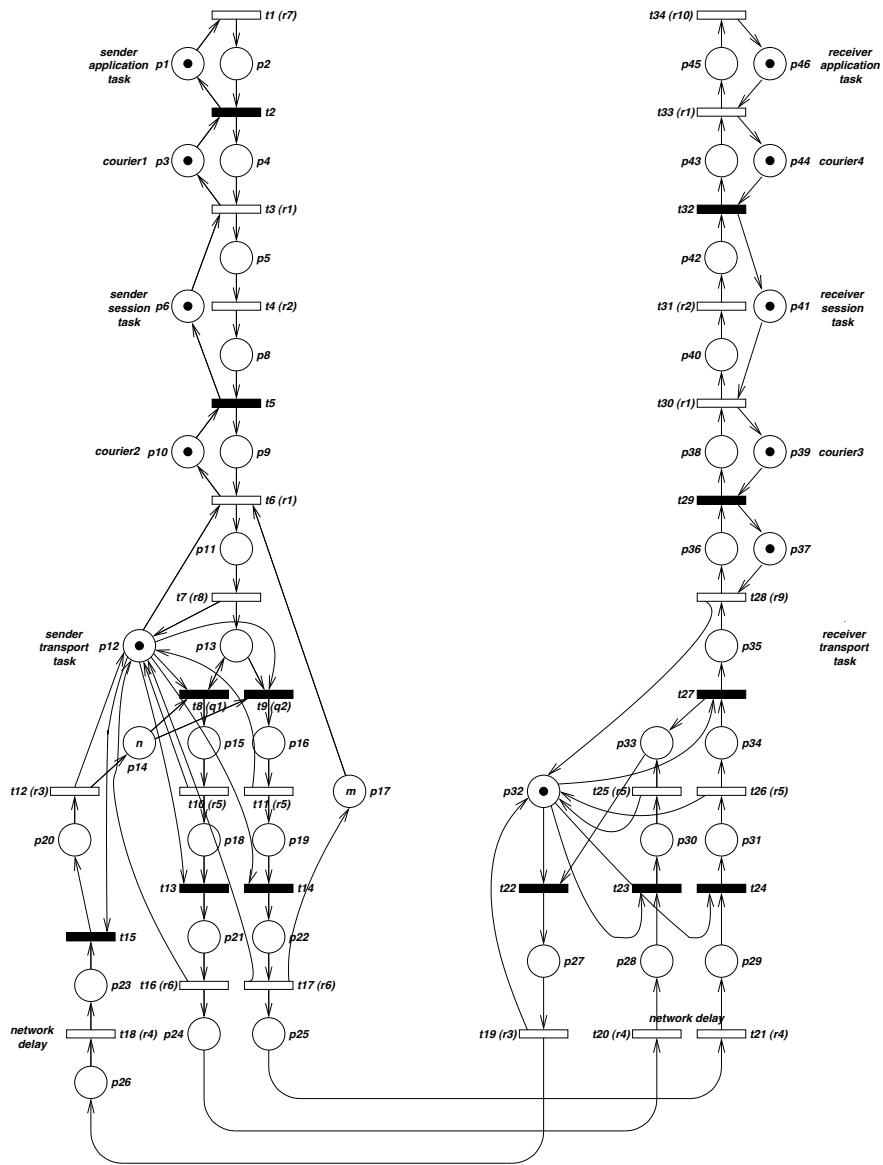


Figure 3: GSPN model of the Courier communications protocol [13].

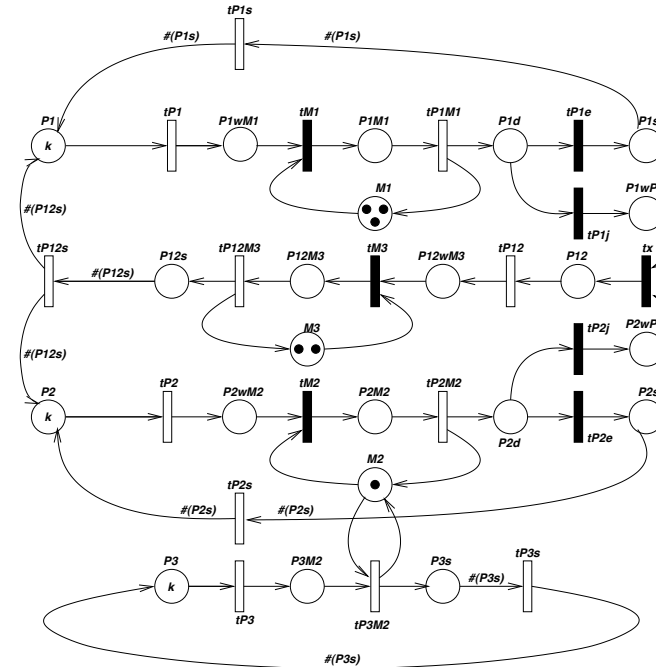


Figure 4: GSPN model of a Flexible Manufacturing System [1].

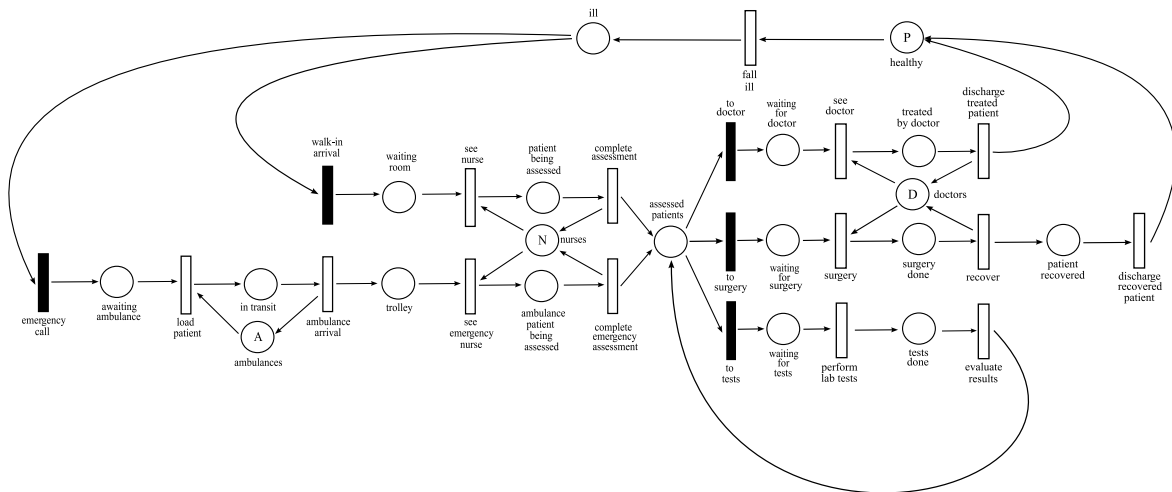


Figure 5: GSPN model of patient flow in a hospital environment.

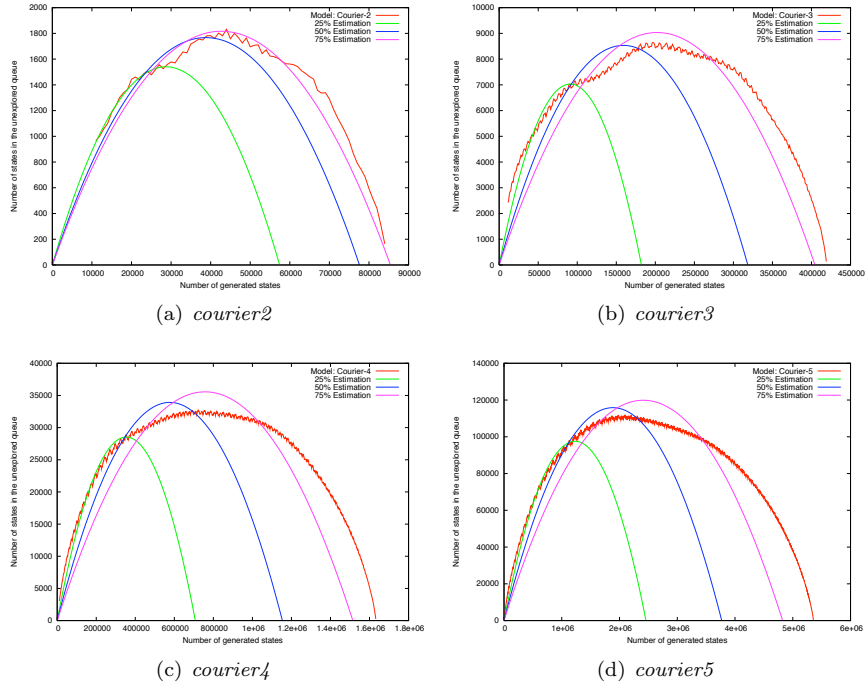


Figure 6: Least-squares fitting estimation of the number of states in the Courier model at 25%, 50% and 75% of the state generation process.

communication protocol [13]. Data flows from a sender (p_1 to p_{26}) to a receiver (p_{27} to p_{46}) via a network. The sender's transport layer fragments outgoing data packets; this is modelled as two paths between p_{13} and p_{35} . The transport layer is characterised by two important parameters: the sliding window size n (p_{14}) and the transport space m (p_{17}). In our investigations we will be varying n to produce varying sizes of state spaces.

Fig. 4 shows a 22-place GSPN model of a flexible manufacturing system [1]. The model describes an assembly line with three types of machines ($M1$, $M2$ and $M3$) which assemble four types of parts ($P1$, $P2$, $P3$ and $P12$). Initially, there are k unprocessed parts of each type $P1$, $P2$ and $P3$ in the system. There are no parts of type $P12$ at start-up since these are assembled from processed parts of type $P1$ and $P2$ by the machines of type $M3$. When parts of any type are finished, they are stored for shipping on places $P1s$, $P2s$, $P3s$ and $P12s$.

Fig. 5 shows a GSPN model of a hospital's Accident and Emergency department. The key parameters in this model are the numbers of patients (P), nurses (N), doctors (D) and ambulances (A).

Table 1 shows the number of states in the underlying CTMCs for the Courier and FMS models in terms of the parameters in the GSPN models. Note that the first column gives a short name for each of the configurations by which we will refer to it for the remainder of this paper. Likewise, Table 2 contains the number of tangible states in the hospital model for various values of P , N , D and A , along with an associated short name in the first column.

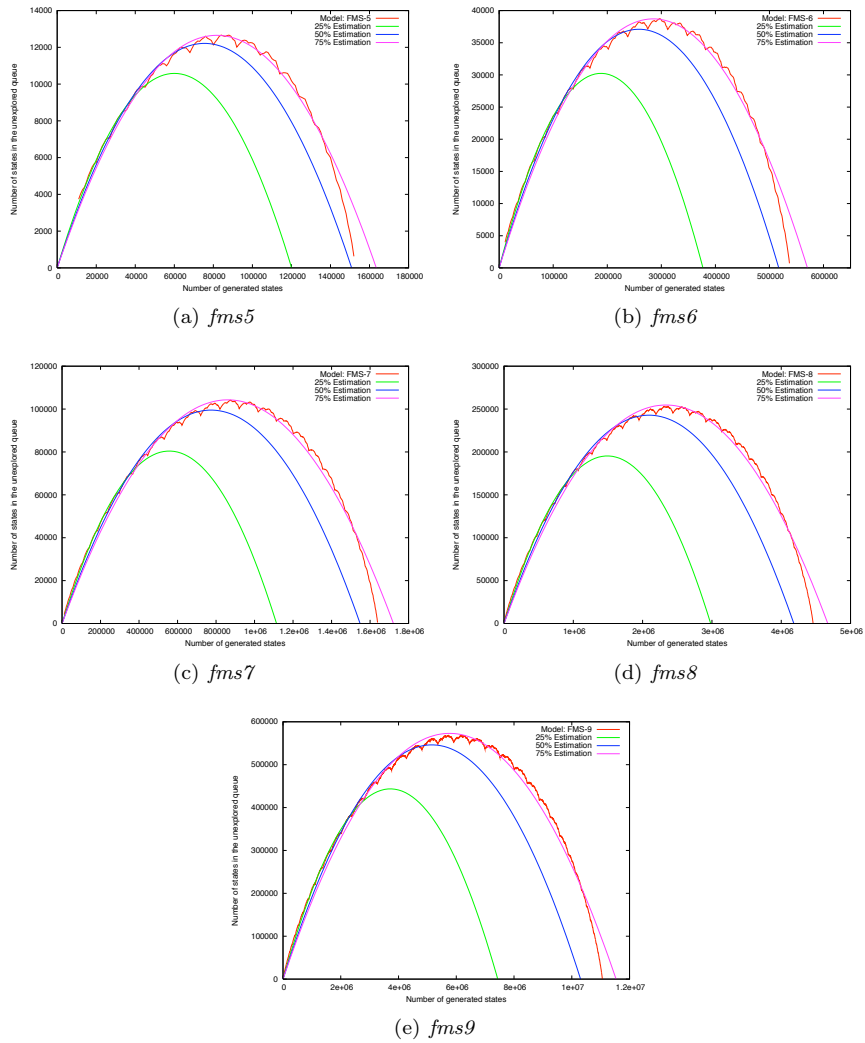


Figure 7: Least-squares fitting estimation of the number of states in the FMS model at 25%, 50% and 75% of the state generation process.

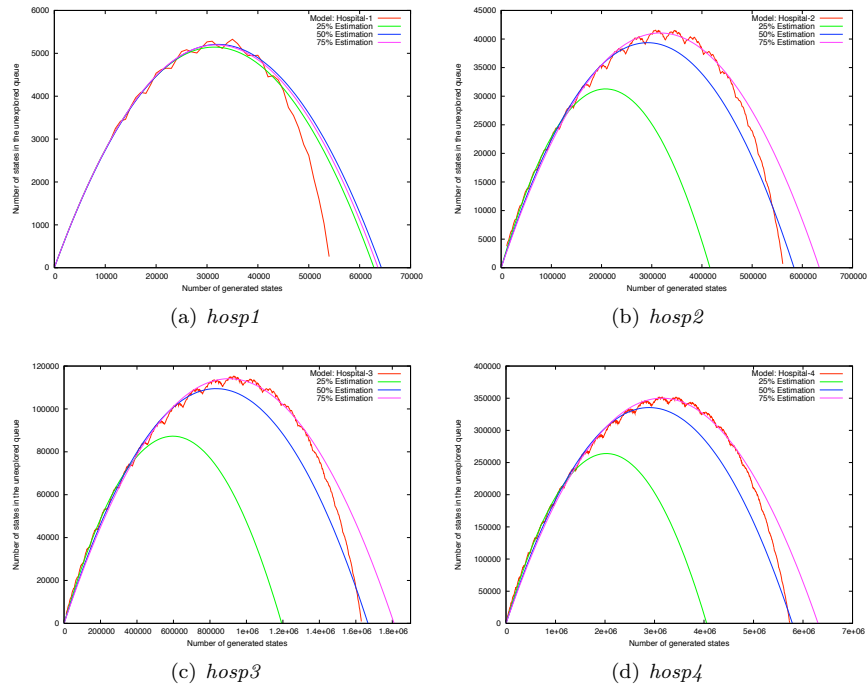


Figure 8: Least-squares fitting estimation of the number of states in the Hospital model at 25%, 50% and 75% of the state generation process.

Model Name	Tangible States	25%		50%		75%	
		Estimate	% Error	Estimate	% Error	Estimate	% Error
<i>courier2</i>	84 600	47 440	43.9%	72 617	14.2%	83 720	1.0%
<i>courier3</i>	419 400	180 880	56.9%	317 845	24.2%	404 244	3.6%
<i>courier4</i>	1 632 600	707 434	56.7%	1 153 623	29.3%	1 514 634	7.2%
<i>courier5</i>	5 358 600	2 458 700	54.1%	3 769 918	29.6%	4 824 976	10.0%
<i>fms5</i>	152 712	116 370	23.8%	149 793	1.9%	163 166	6.8%
<i>fms6</i>	537 768	375 743	30.1%	517 008	3.9%	570 206	6.0%
<i>fms7</i>	1 639 440	1 113 737	32.1%	1 548 048	5.6%	1 721 391	5.0%
<i>fms8</i>	4 459 455	2 982 118	33.1%	4 181 830	6.2%	4 671 613	4.8%
<i>fms9</i>	11 058 190	7 435 833	32.8%	10 304 305	6.8%	11 532 975	4.3%
<i>hosp1</i>	54 228	53 630	1.1%	61 825	14.0%	63 015	16.2%
<i>hosp2</i>	561 704	414 694	26.2%	582 222	3.7%	633 916	12.9%
<i>hosp3</i>	1 630 905	1 193 169	26.8%	1 666 353	2.2%	1 809 587	11.0%
<i>hosp4</i>	5 728 971	4 053 685	29.2%	5 784 496	1.0%	6 303 895	10.0%
Average error			34.4%		11.0%		7.6%

Table 3: Difference between the number of states predicted by least-squares fitting and the actual number generated. Results are presented at three points in the state generation process for each of the three GSPN models.

The results of the estimation process using least-squares fitting are shown graphically in Figs. 6, 7 and 8 for each of the three GSPN models. The estimation was performed at three points in the state generation process where 25%, 50% and 75% of the total number of states had been generated. The number of states predicted at each of these points for the three models, along with the percentage error compared with the actual final amount, is given in Table 3.

We observe that early on in the state generation process the estimation of the total number of states varies greatly from the actual number in all but the smallest Hospital model (*hosp1*), with an average percentage error of 34.4%. By the half-way stage, however, the estimate is usually much more accurate (all within 7% in the case of the FMS model) and the average error falls to 11.0%. When three-quarters of the state space has been generated the estimation further improves (with an average overall error of 7.6%), although the accuracy does decrease compared with the half-way point estimate in the Hospital model. Nevertheless, we believe that the ability to estimate to within approximately 10% of the actual total number of states by the half-way point in the generation process demonstrates the applicability of our technique.

5 Conclusion

We have demonstrated how least-squares fitting can be used to accurately estimate the total number of states in the underlying CTMCs of high-level models during the state generation process. Results from our experiments suggest that estimates produced in this way do provide a good guide to the likely eventual number of states. On average, an error of 11.0% in the predicted total was observed at the half-way point in the state generation process.

With the accuracy of our technique demonstrated we will now investigate incorporating it into DNAmaca and derived tools such as HYDRA [3]. In particular, it would be interesting to employ the dynamic process management features of the MPI-2 parallel programming library [6], in conjunction with our estimation method and a parallel state-space generator [8], to automatically spawn extra processors when analysing large models.

References

- [1] G. Ciardo and K.S. Trivedi. A decomposition approach for stochastic reward net models. *Performance Evaluation*, 18(1):37–59, 1993.
- [2] D.D. Deavours and W.H. Sanders. “On-the-fly” solution techniques for stochastic Petri nets and extensions. *IEEE Transactions on Software Engineering*, 24(10):889–902, 1998.
- [3] N.J. Dingle. *Parallel Computation of Response Time Densities and Quantiles in Large Markov and Semi-Markov Models*. PhD thesis, Imperial College, London, United Kingdom, 2004.
- [4] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *GNU Scientific Library: Reference Manual*. Network Theory Ltd., February 2003.

- [5] G.H. Golub and C.F. van Loan. *Matrix Computations*. John Hopkins University Press, 3rd edition, 1996.
- [6] W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2: Advanced features of the Message-Passing Interface*. MIT Press, Cambridge, Massachusetts, 1999.
- [7] W.J. Knottenbelt. Generalised Markovian analysis of timed transition systems. Master's thesis, University of Cape Town, Cape Town, South Africa, July 1996.
- [8] W.J. Knottenbelt. *Parallel Performance Analysis of Large Markov Models*. PhD thesis, Imperial College, London, United Kingdom, February 2000.
- [9] W.J. Knottenbelt, P.G. Harrison, M.A. Mestern, and P.S. Kritzinger. A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation*, 39(1–4):127–148, February 2000.
- [10] R. Pelánek and P. Šimeček. Estimating state space parameters. In *PDMC'08, Proc. 7th Intl. Workshop on Parallel and Distributed Methods in Verification*, Budapest, Hungary, March 2008. Elsevier.
- [11] E. Tronci, G. Della Penna, B. Intrigila, and M.V. Zilli. A probabilistic approach to automatic verification of concurrent systems. In *Proc. 8th IEEE Asia-Pacific Software Engineering Conference (APSEC 2001)*.
- [12] J.F. Watson III and A.A. Desrochers. State-space size estimation of Petri nets: A bottom-up perspective. *IEEE Transactions on Robotics and Automation*, 10(4):555–561, August 1994.
- [13] C.M. Woodside and Y. Li. Performance Petri net analysis of communication protocol software by delay-equivalent aggregation. In *Proceedings of the 4th International Workshop on Petri nets and Performance Models (PNPM'91)*, pages 64–73, Melbourne, Australia, 2–5 December 1991. IEEE Computer Society Press.

Hypercube Communication Structures Analysis via Parametric Petri Nets^{*}

Zaitsev D.A.¹

Shmeleva T.R.²

Abstract

A model of a hypercube communication structure of an arbitrary size with an arbitrary number of dimensions in the form of parametric Petri net was constructed. A technique of the linear invariants calculation for parametric Petri nets was developed which allowed the analysis of transmissions involving an arbitrary number of communicating devices (routers) in hypercube. The compulsory buffering of the packets inevitably leads to possible blockings of communicating devices. The structure of complex deadlocks involving an arbitrary number of communicating devices caused by both the chain (cycle) of blockings and the isolation was studied. In real-life networks, described deadlocks lead to considerable decrease of performance.

1 Introduction

As the number of communicating devices and the structure of a network are varying considerably for real-life networks, a technique is required that could manage an arbitrary number of devices constituting an arbitrary structure.

Ajmoné Marsan [1] started studying Petri nets composed as a repetition of a basic component for the performance evaluation of CSMA/CD bus LAN. The linear structure was built but the technique of its analysis for an arbitrary number of components was not presented. In [2] the parametric composition of functional Petri nets [3] was applied for the analysis of linear structures of communicating devices. A simpler direct approach [4] was applied to tree-like structures for the verification of switched Ethernet protocols.

The goal of the present paper is the generalization of the approach described in [4] on hypercube communication structure of an arbitrary size with an arbitrary number of dimensions. In [4] the type Petri net model of Ethernet switch with the compulsory buffering of frames was built and used for the composition of tree-like communication structure. As far as routing and switching tables are not represented in the model [4], the same model can describe both switches and routers. We use its modification as the model of a generalized communication device for the composition of hypercube communication structures. The only difference is the number of ports and the disposition of ports on the facets of a hypercube for the further composition of hypercube communication structures (HCS). The direction for future work is the generalization of the obtained results for an arbitrary structure and the modeling of cut-through devices as well.

For the analysis of parametric Petri nets properties linear invariants [5] are used. According to [6,7], an ideal model of a communication system should be bounded, safe and live Petri net. P-invariants allow the proof of boundedness and safeness. But t-invariants do not allow the proof of liveness. That is why special graphs are introduced to prove that the model is not live via deadlocks observation.

^{*} Work supported by NATO grant ICS CLG 982698

¹ Odessa National Academy of Telecommunications, Ukraine, zsoftua@yahoo.com

² Odessa National Academy of Telecommunication, Ukraine, tishtri@rambler.ru

2 Model of Communication Device

Communication devices of the packet-switching networks, such as switches and routers, consist of a finite number of ports and implement the forwarding of the arrived (from a port) packet to the destination port. Packet header information and switch/router address table are used for the calculation of the destination port number. The device implements either the compulsory buffering of packets into its internal buffer or cut-through possibilities. Even cut-through devices employ the buffering when the destination port is busy. Ports work in full-duplex mode providing two channels: for the receiving and the sending of packets; moreover, ports have their own buffers for each channel with the capacity available for the storing of one packet usually.

In the present work we abstract from packet headers and address tables and consider devices with the compulsory buffering only. All the capacities are measured in number of packets.

Let us consider a d dimension space, where $d = 1, 2, \dots$. Each communication device is represented by the hypercube of the size 1 in d dimension space. The communication structure composed by connected communication devices constitutes the hypercube of the size k , where $k = 1, 2, \dots$. So the total number of devices is $N_{dev} = k^d$. Each device R^{i_1, i_2, \dots, i_d} has its index (i_1, i_2, \dots, i_d) , where $i_u = \overline{1, k}$, $u = \overline{1, d}$. The model of the hypercube communication structure is denoted as $H_{d, k}$. Further we describe the Petri net model of a device and then the composition of a communication structure model via connections of a device with its neighbors.

The model of a hypercube device is denoted as $H_{d, 1}$ (number of dimensions equals to d , size of the structure equals to 1). On each facet of a hypercube device R^{i_1, i_2, \dots, i_d} in d -dimension space a port is situated. So each device has $N_{port} = 2 \cdot d$ ports; two ports for each dimension are situated at the opposite facets of the hypercube. To denote opposite facets for a dimension j ($j = \overline{1, d}$) the number of the direction is used. The direction is denoted by the variable n ; the value $n = 1$ is used for the direction to zero in the corresponding dimension and the value $n = 2$ is used for the opposite direction to infinity. So the ports may be denoted with the following indices $port_{j, n}^{i_1, i_2, \dots, i_d}$, where $i_u = \overline{1, k}$, $u = \overline{1, d}$, $j = \overline{1, d}$, $n = 1, 2$. Each port is represented by the two channels (input, output), each channel is represented by a pair of places: one place for the packets buffer, the other – for the buffer capacity. So each port of the device R^{i_1, i_2, \dots, i_d} is represented by the four following contact places:

- $pi_{j, n}^{i_1, i_2, \dots, i_d}$ - input buffer of packets;
- $pil_{j, n}^{i_1, i_2, \dots, i_d}$ - capacity of input buffer (equals to 1);
- $po_{j, n}^{i_1, i_2, \dots, i_d}$ - output buffer of packets;
- $pol_{j, n}^{i_1, i_2, \dots, i_d}$ - capacity of output buffer (equals to 1).

The inside of the device contains $N_{port} + 1$ following places. The packets redirected to the port $port_{j, n}^{i_1, i_2, \dots, i_d}$ are stored in the corresponding place $pb_{j, n}^{i_1, i_2, \dots, i_d}$

and one place $pbl^{i_1, i_2, \dots, i_d}$ contains the capacity of the internal buffer, where $j' = \overline{1, d}$, $n' = 1, 2$. Notice that the internal buffer is represented by the set of places $pb_{j', n'}^{i_1, i_2, \dots, i_d}$ (one place for each port) to distinguish the number of the destination port given by indices j', n' .

The transitions of the device R^{i_1, i_2, \dots, i_d} provide the redirection of the input packets from an input port buffer place $pi_{j, n}^{i_1, i_2, \dots, i_d}$ into one of the internal buffer places $pb_{j', n'}^{i_1, i_2, \dots, i_d}$, $j' \neq j, n' \neq n$ and then the transmission of the packets from the internal buffer place $pb_{j', n'}^{i_1, i_2, \dots, i_d}$ to the output buffer of the target port $po_{j', n'}^{i_1, i_2, \dots, i_d}$. Moreover, the limitations of buffers capacities should be taken into consideration: check and decrease the buffer size at putting the packet into the buffer; increase the buffer size at getting the packet from the buffer. So each port $port_{j, n}^{i_1, i_2, \dots, i_d}$ of the device R^{i_1, i_2, \dots, i_d} is supplied by $N_{port} = (N_{port} - 1) + 1$ following transitions:

- one transition for the output channel $to_{j, n}^{i_1, i_2, \dots, i_d}$ with the input arcs from places $pb_{j, n}^{i_1, i_2, \dots, i_d}$, $pol_{j, n}^{i_1, i_2, \dots, i_d}$ and the output arcs to places $po_{j, n}^{i_1, i_2, \dots, i_d}$, $pbl_{j, n}^{i_1, i_2, \dots, i_d}$;
- $N_{port} - 1$ transitions $ti_{j, n, j', n'}^{i_1, i_2, \dots, i_d}$, $j' = \overline{1, d}$, $n' = 1, 2$, $j' \neq j$, $n' \neq n$ for the input channel with the input arcs from places $pi_{j, n}^{i_1, i_2, \dots, i_d}$, $pbl_{j, n}^{i_1, i_2, \dots, i_d}$ and the output arcs to places $pb_{j', n'}^{i_1, i_2, \dots, i_d}$, $pil_{j, n}^{i_1, i_2, \dots, i_d}$.

The formal parametric description of the net $H_{d, 1}$ is the following:

$$\left(\left(\left(ti_{j, n, j', n'}^{i_1, i_2, \dots, i_d} : pi_{j, n}^{i_1, i_2, \dots, i_d}, pbl_{j, n}^{i_1, i_2, \dots, i_d} \rightarrow pb_{j', n'}^{i_1, i_2, \dots, i_d}, pil_{j, n}^{i_1, i_2, \dots, i_d} \right), \right. \right. \\ \left. \left. \left(to_{j, n}^{i_1, i_2, \dots, i_d} : pb_{j, n}^{i_1, i_2, \dots, i_d}, pol_{j, n}^{i_1, i_2, \dots, i_d} \rightarrow po_{j, n}^{i_1, i_2, \dots, i_d}, pbl_{j, n}^{i_1, i_2, \dots, i_d} \right) \right), j = \overline{1, d}, n = \overline{1, 2} \right) \quad (1)$$

If the net $H_{d, 1}$ is considered as a model of the device R^{i_1, i_2, \dots, i_d} in the hypercube structure, the upper indices of its hypercube cell should be added:

$$\left(\left(\left(ti_{j, n, j', n'}^{i_1, i_2, \dots, i_d} : pi_{j, n}^{i_1, i_2, \dots, i_d}, pbl_{j, n}^{i_1, i_2, \dots, i_d} \rightarrow pb_{j', n'}^{i_1, i_2, \dots, i_d}, pil_{j, n}^{i_1, i_2, \dots, i_d} \right), \right. \right. \\ \left. \left. \left(to_{j, n}^{i_1, i_2, \dots, i_d} : pb_{j, n}^{i_1, i_2, \dots, i_d}, pol_{j, n}^{i_1, i_2, \dots, i_d} \rightarrow po_{j, n}^{i_1, i_2, \dots, i_d}, pbl_{j, n}^{i_1, i_2, \dots, i_d} \right) \right), j = \overline{1, d}, n = \overline{1, 2} \right).$$

The notation of the transition connections in the form

$$t : px_1, \dots, px_u \rightarrow py_1, \dots, py_v$$

is widely used for Petri nets, for instance, in Tina[8] software. It means that transition t has input arcs from places px_1, \dots, px_u and output arcs to places py_1, \dots, py_v , where u, v are the numbers of input and output places correspondingly. If necessary, the multiplicities of arcs are added using the notation $p_i * l$, where l is the multiplicity of the corresponding arc connecting p_i and t .

The net represented by (1) is called parametric Petri net because its description has the parameter d for the calculation of the elements indices. The size of the net $H_{d,1}$ is unlimited and represented by the parameter d .

The parametric model $H_{d,1}$ is illustrated by the examples for the following concrete number of dimensions $d = 2, 3$ shown in Fig. 1. It is rather difficult to visualize models for larger numbers of dimensions.

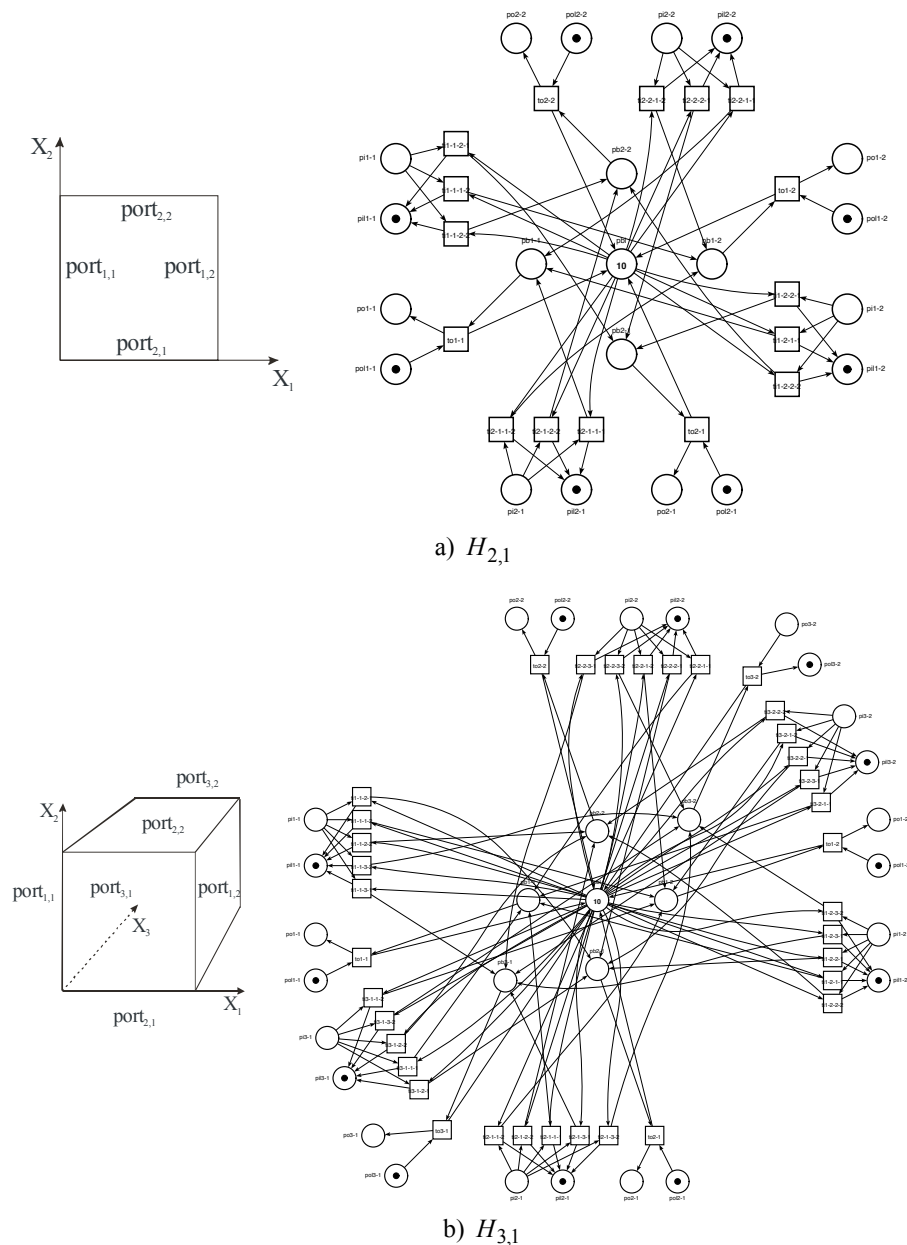


Fig. 1. Examples of parametric model $H_{d,1}$ for $d = 2, 3$.

3 P-invariants of Communication Device Model

Using the parametric description (1) of the communication device model $H_{d,1}$ given in the previous section the following system was constructed for the calculation of p-invariants:

$$\begin{cases} to_{j,n} : xpb_{j,n} + xpol_{j,n} = xpo_{j,n} + xpb_l, \\ ti_{j,n,j',n'} : xpi_{j,n} + xpb_l = xpb_{j',n'} + xpil_{j,n}, \\ j = \overline{1,d}, n = 1,2, j' = \overline{1,d}, n' = 1,2, j' \neq j, n' \neq n. \end{cases} \quad (2)$$

Notice that the system (2) has parametric form; its parameter is the number of dimensions d . System was constructed directly on the description (1) using the usual rule [5] that each equation corresponds to transition and contains sums for its input and output arcs, which are equal. Sums should be calculated using the multiplicities of arcs but all the arcs of (1) have the multiplicity equaling to unit.

The total number of system (2) equations is $N_{d,1}^t = 4 \cdot d^2$.

The total number of system (2) variables is $N_{d,1}^p = 10 \cdot d + 1$.

To study p-invariants of the model for any number of dimensions the system (2) should be solved in the parametric form. The obtained parametric solution of the system (2) has the following form:

$$\begin{pmatrix} (pi_{j,n}, pil_{j,n}), j = \overline{1,d}, n = 1,2; \\ (po_{j,n}, pol_{j,n}), j = \overline{1,d}, n = 1,2; \\ (pbl, (pb_{j,n}, j = \overline{1,d}, n = 1,2)) \\ ((pb_{j,n}, pi_{j,n}, po_{j,n}), j = \overline{1,d}, n = 1,2) \\ (pbl, ((pil_{j,n}, pol_{j,n}), j = \overline{1,d}, n = 1,2)) \end{pmatrix} \quad (3)$$

The way of the solutions description is common enough for sparse vectors and especially for the Petri net theory. Only nonzero components are mentioned by the names of the corresponding places. The nonzero multiplier 1 is omitted; in case it is not the unit, the notation $p * x$ is used where x is the value of the invariant for place p . Such notation is adopted in the Tina software [8] which was used for obtaining the Petri net figures in this paper. A line of the matrix (3) gives us a set of lines according to the used indices i, j, n except the last two lines which contain variable number of components given by indices.

A heuristic algorithm was employed for the construction of the matrix (3) but further the proof is presented that (3) is a solution of (1). The fact that (3) is the basis solution is not required for the conclusion about p-invariance of $H_{d,1}$.

The total number of solutions in the matrix (3) is $N_{d,1}^{pinv} = 4d + 3$.

Lemma 1. Each line of the matrix (3) is a solution of the system (2).

Proof. Let us substitute each parametric line of (3) into each parametric equation of the system (2). It gives us the correct statement. At the substitution, the different names of indices are chosen. For instance, let us substitute the fourth line of (3)

$$((pb_{l,m}, pi_{l,m}, po_{l,m}), l = \overline{1,d}, m = 1,2)$$

into the second equation of (2)

$$xpi_{j,n} + xpb_l = xpb_{j',n'} + xpil_{j,n}, j = \overline{1,d}, n = 1,2, j' = \overline{1,d}, n' = 1,2, j' \neq j, n' \neq n.$$

For each concrete equation given by valid tuple (j, n, j', n') the solution contains $pi_{j,n}$ at $l = j, m = n$ and $pb_{j',n'}$ at $l = j', m = n'$, moreover the other variables of the equation $xpbl, xpil_{j',n'}$ are not mentioned in the solution. So we obtain:

$1+0=1+0$ and further $1=1$
for each equation.

The first two solutions of (3) make slight difference: they represent series of lines given by their indices. Let us substitute the first parametric line of (3)

$$(pi_{l,m}, pil_{l,m}), \quad l = \overline{1, d}, \quad m = 1, 2;$$

into the second parametric equation of (2)

$$xpi_{j,n} + xpbl = xpb_{j',n'} + xpil_{j,n}, \quad j = \overline{1, d}, \quad n = 1, 2, \quad j' = \overline{1, d}, \quad n' = 1, 2, \quad j' \neq j, \quad n' \neq n.$$

We obtain:

- when $l \neq j$ or $m \neq n$: $0+0=0+0$ and further $0=0$;
- when $l = j$ and $m = n$: $1+0=0+1$ and further $1=1$.

In the same way all the 5×2 combinations are checked. ■

Theorem 1. The net $H_{d,1}$ is a p-invariant Petri net for an arbitrary natural number d .

Proof. Let us consider the sum of the fourth and the fifth lines of the matrix (3) which represent the solutions of the system (2) according to Lemma 1:

$$((pb_{j,n}, pi_{j,n}, po_{j,n}), \quad j = \overline{1, d}, \quad n = 1, 2)$$

plus

$$(pbl, ((pil_{j,n}, pol_{j,n}), \quad j = \overline{1, d}, \quad n = 1, 2))$$

equals to

$$(pbl, ((pil_{j,n}, pol_{j,n}, pb_{j,n}, pi_{j,n}, po_{j,n}), \quad j = \overline{1, d}, \quad n = 1, 2)) \quad (4)$$

As all the $N_{d,1}^p = 10 \cdot d + 1$ places are mentioned in this invariant, the net $H_{d,1}$ is a p-invariant Petri net for an arbitrary natural number d . Moreover, as each component of (4) equals to the unit, the net $H_{d,1}$ is a safe and bounded Petri net for an arbitrary natural number d . ■

4 Composition of HCS

The connections of communication devices in the hypercube are provided by the fusion (union) of the corresponding contact places of neighbor devices.

Let us consider an internal communication device $R^{i_1, \dots, i_j, \dots, i_d}$, $i_u = \overline{2, k-1}$, $u = \overline{1, d}$, $j = \overline{1, d}$:

- places of $port_{j,1}^{i_1, \dots, i_j, \dots, i_d}$ are fused with the corresponding places of $port_{j,2}^{i_1, \dots, i_j-1, \dots, i_d}$, device $R^{i_1, \dots, i_j-1, \dots, i_d}$ in such a way that place $po_{j,1}^{i_1, \dots, i_j, \dots, i_d}$ is fused with $pi_{j,2}^{i_1, \dots, i_j-1, \dots, i_d}$, place $pol_{j,1}^{i_1, \dots, i_j, \dots, i_d}$ – with $pil_{j,2}^{i_1, \dots, i_j-1, \dots, i_d}$, place $pi_{j,1}^{i_1, \dots, i_j, \dots, i_d}$ – with $po_{j,2}^{i_1, \dots, i_j-1, \dots, i_d}$, place $pil_{j,1}^{i_1, \dots, i_j, \dots, i_d}$ – with $pol_{j,2}^{i_1, \dots, i_j-1, \dots, i_d}$;

– places of $port_{j,2}^{i_1,\dots,i_j,\dots,i_d}$ are fused with the corresponding places of $port_{j,1}^{i_1,\dots,i_j+1,\dots,i_d}$, device $R^{i_1,\dots,i_j+1,\dots,i_d}$ in such a way that place $po_{j,2}^{i_1,\dots,i_j,\dots,i_d}$ is fused with $pl_{j,1}^{i_1,\dots,i_j+1,\dots,i_d}$, place $pol_{j,2}^{i_1,\dots,i_j,\dots,i_d}$ – with $pil_{j,1}^{i_1,\dots,i_j+1,\dots,i_d}$, place $pl_{j,2}^{i_1,\dots,i_j,\dots,i_d}$ – with $po_{j,1}^{i_1,\dots,i_j+1,\dots,i_d}$, place $pil_{j,2}^{i_1,\dots,i_j,\dots,i_d}$ – with $pol_{j,1}^{i_1,\dots,i_j+1,\dots,i_d}$.

To avoid duplicity, the names of the places for the zero direction ports $n=1$ will be considered with respect to the current device, for the infinity direction ports $n=2$ – with respect to the neighbor devices and their zero direction ports $n=1$. So the names of the fusion places have only the indices of the zero direction ports $n=1$. Moreover, to simplify further notations, the places with the indices of the infinity direction ports $n=2$ on the facets (borders) of the communication hypercube are named with respect to non existing devices with the indices equaling to $k+1$. So the names of ports with the indices of the infinity direction $n=2$ do not appear in the hypercube. The communication hypercube structure described above is denoted as $H_{d,k}$. An example of $H_{d,k}$ for $d=3, k=4$ is represented in Fig. 2.

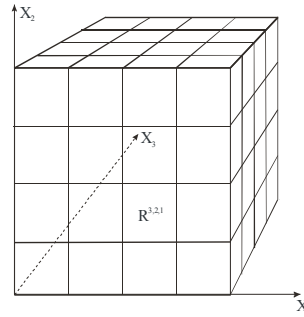


Fig. 2. Scheme of the communication structure $H_{3,4}$

The formal description of $H_{d,k}$ composition is given with the following:

$$\left(\begin{array}{l} pl_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := po_{j,2}^{i_1,\dots,i_j,\dots,i_d} \cup pl_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} \\ pil_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := pol_{j,2}^{i_1,\dots,i_j,\dots,i_d} \cup pil_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} \\ po_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := pl_{j,2}^{i_1,\dots,i_j,\dots,i_d} \cup po_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} \\ pol_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := pil_{j,2}^{i_1,\dots,i_j,\dots,i_d} \cup pol_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} \end{array} \right), i_u = \overline{1, k-1}, u = \overline{1, d}, j = \overline{1, d} \right,$$

$$\left(\begin{array}{l} pl_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := po_{j,2}^{i_1,\dots,i_j,\dots,i_d} \\ pil_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := pol_{j,2}^{i_1,\dots,i_j,\dots,i_d} \\ po_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := pl_{j,2}^{i_1,\dots,i_j,\dots,i_d} \\ pol_{j,1}^{i_1,\dots,i_j+1,\dots,i_d} := pil_{j,2}^{i_1,\dots,i_j,\dots,i_d} \end{array} \right), i_u = \overline{1, k-1}, u = \overline{1, d}, j = \overline{1, d}, u \neq j, i_j = k \right).$$

The union sign \cup denotes the fusion of places; the left column gives new names of places.

5 P-invariants of HCS

Using the abstract description of the communication hypercube model $H_{d,k}$ given in the previous section, the following system is constructed for the calculation of p-invariants:

$$\begin{cases} to_{j,1}^{i_1, \dots, i_d} : xpb_{j,1}^{i_1, \dots, i_d} + xpol_{j,1}^{i_1, \dots, i_d} = xpo_{j,1}^{i_1, \dots, i_d} + xpb_{j,1}^{i_1, \dots, i_d}, \\ ti_{j,1,j',n'}^{i_1, \dots, i_d} : xpi_{j,1}^{i_1, \dots, i_d} + xpb_{j,1}^{i_1, \dots, i_d} = xpb_{j',n'}^{i_1, \dots, i_d} + xpi_{j,1}^{i_1, \dots, i_d}, \\ to_{j,2}^{i_1, \dots, i_d} : xpb_{j,2}^{i_1, \dots, i_d} + xpi_{j,1}^{i_1, \dots, i_d+1} = xpi_{j,1}^{i_1, \dots, i_d+1} + xpb_{j,2}^{i_1, \dots, i_d}, \\ ti_{j,2,j',n'}^{i_1, \dots, i_d} : xpo_{j,1}^{i_1, \dots, i_d+1} + xpb_{j,1}^{i_1, \dots, i_d} = xpb_{j',n'}^{i_1, \dots, i_d} + xpo_{j,1}^{i_1, \dots, i_d+1}, \\ j = \overline{1, d}, \quad j' = \overline{1, d}, \quad n' = 1, 2, \quad j' \neq j, \quad n' \neq n, \quad i_u = \overline{1, k}, \quad u = \overline{1, d}. \end{cases} \quad (5)$$

The total number of system (5) equations is $N_{d,k}^t = 4 \cdot d^2 \cdot k^d$.

The total number of system (5) variables is $N_{d,k}^p = (6 \cdot d + 1) \cdot k^d + 4 \cdot d \cdot k^{d-1}$.

The obtained parametric solution has the following form:

$$\left\{ \begin{aligned} &(pi_{j,1}^{i_1, \dots, i_d}, pil_{j,1}^{i_1, \dots, i_d}), \quad j = \overline{1, d}, \quad (i_u = \overline{1, k}, \quad u = \overline{1, d}, \quad u \neq j), \quad i_j = \overline{1, k+1}; \\ &(po_{j,1}^{i_1, \dots, i_d}, pol_{j,1}^{i_1, \dots, i_d}), \quad j = \overline{1, d}, \quad (i_u = \overline{1, k}, \quad u = \overline{1, d}, \quad u \neq j), \quad i_j = \overline{1, k+1}; \\ &(pbl_{j,n}^{i_1, \dots, i_d}, (pb_{j,n}^{i_1, \dots, i_d}, j = \overline{1, d}, \quad n = 1, 2)), \quad i_u = \overline{1, k}, \quad u = \overline{1, d}; \\ &((pb_{j,n}^{i_1, \dots, i_d}, n = 1, 2, j = \overline{1, d}, i_u = \overline{1, k}, u = \overline{1, d}), (pi_{j,1}^{i_1, \dots, i_d}, po_{j,1}^{i_1, \dots, i_d}), j = \overline{1, d}, i_u = \overline{1, k}, u = \overline{1, d}), \\ &((pi_{j,1}^{i_1, \dots, i_d}, po_{j,1}^{i_1, \dots, i_d}), j = \overline{1, d}, \quad i_u = \overline{1, k}, \quad u = \overline{1, d}, \quad u \neq j, \quad i_j = k+1) \\ &((pbl_{j,n}^{i_1, \dots, i_d}, (pil_{j,1}^{i_1, \dots, i_d}, pol_{j,1}^{i_1, \dots, i_d}), j = \overline{1, d}), \quad i_u = \overline{1, k}, \quad u = \overline{1, d}), \\ &((pil_{j,1}^{i_1, \dots, i_d}, pol_{j,1}^{i_1, \dots, i_d}), j = \overline{1, d}, \quad i_u = \overline{1, k}, \quad u = \overline{1, d}, \quad u \neq j, \quad i_j = k+1) \end{aligned} \right\} \quad (6)$$

The total number of solutions is $N_{d,k} = (1 + 2 \cdot d) \cdot k^d + 2 \cdot d \cdot k^{d-1} + 2$.

Lemma 2. Each line of the matrix (6) is a solution of the system (5).

Theorem 3. The net $H_{d,k}$ is a p-invariant Petri net for arbitrary natural numbers d, k .

The proofs of Lemma 2 and Theorem 3 were done in the same way as for the net $H_{d,1}$ (Section 3).

6 Adding Models of Terminal Devices

The communication devices are attached to each other constituting a communication structure but they are created only for the packets transmission among the terminal devices: workstations and servers. In the present work, the client-server technique of interaction is not studied, so the types of terminal devices are not distinguished. An abstract terminal device provides at least two basic functions: send packet and receive packet. These basic functions are implemented in the model represented in Fig. 3.

The model contains an internal buffer of the packets qb ; transition si models the receiving of the packets, while transition so models the sending. The model keeps the balance of input and output packets; the limitation of buffer qb size is not considered.

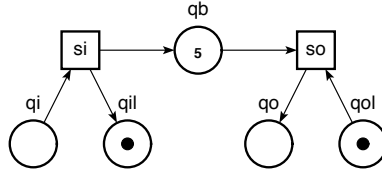


Fig. 3. Petri net model of a terminal device

The terminal devices shown in Fig. 3 are attached to the border ports of the hypercube structure; the corresponding model is denoted as $HT_{d,k}$. The terminal device in hypercube structure is denoted as $A^{i_1, \dots, i_j, \dots, i_d}$, where $i_u = \overline{1, k}, u = \overline{1, d}, j = \overline{1, d}, u \neq j, i_j \in \{1, k\}$ and attached to the communication device $R^{i_1, \dots, i_j, \dots, i_d}$. The formal description of $HT_{d,k}$ composition using $HT_{d,k}$, $A^{i_1, \dots, i_j, \dots, i_d}$ is given with the following:

$$\left(\begin{array}{l} pi_{j,1}^{i_1, \dots, i_j, \dots, i_d} := qo^{i_1, \dots, i_j, \dots, i_d} \cup pi_{j,1}^{i_1, \dots, i_j, \dots, i_d} \\ pil_{j,1}^{i_1, \dots, i_j, \dots, i_d} := qol^{i_1, \dots, i_j, \dots, i_d} \cup pil_{j,1}^{i_1, \dots, i_j, \dots, i_d} \\ po_{j,1}^{i_1, \dots, i_j, \dots, i_d} := qi^{i_1, \dots, i_j, \dots, i_d} \cup po_{j,1}^{i_1, \dots, i_j, \dots, i_d} \\ pol_{j,1}^{i_1, \dots, i_j, \dots, i_d} := qil^{i_1, \dots, i_j, \dots, i_d} \cup pol_{j,1}^{i_1, \dots, i_j, \dots, i_d} \end{array} \right), i_u = \overline{1, k}, u = \overline{1, d}, j = \overline{1, d}, u \neq j, i_j = 1, \\ \left(\begin{array}{l} pi_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} := qi^{i_1, \dots, i_j, \dots, i_d} \cup pi_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} \\ pil_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} := qil^{i_1, \dots, i_j, \dots, i_d} \cup pil_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} \\ po_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} := qo^{i_1, \dots, i_j, \dots, i_d} \cup po_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} \\ pol_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} := qol^{i_1, \dots, i_j, \dots, i_d} \cup pol_{j,1}^{i_1, \dots, i_j+1, \dots, i_d} \end{array} \right), i_u = \overline{1, k}, u = \overline{1, d}, j = \overline{1, d}, u \neq j, i_j = k.$$

In the same way as in Section 5 it was proven that $HT_{d,k}$ is a p-invariant Petri net for any given natural numbers d and k .

7 T-invariants and Deadlocks of HCS

For the calculation of t-invariants the same approach is applied. The only difference is that for t-invariants each equation corresponds to place and variables correspond to transitions. It gains us that the Petri net $H_{d,k}$ is not t-invariant, but it is quite trivial because the modeled system is open as the terminal devices are not attached. It was proven that the model of closed system with attached terminal devices $HT_{d,k}$ is a t-invariant Petri net for arbitrary natural numbers d , k . But the consistency of the model does not imply its liveness.

Each pair of neighbor communication devices can fall into a local deadlock, for instance, when the device $R^{i_1, \dots, i_j, \dots, i_d}$ got l packets directed to the device $R^{i_1, \dots, i_j+1, \dots, i_d}$ and the device $R^{i_1, \dots, i_j+1, \dots, i_d}$ got l packets directed to the device $R^{i_1, \dots, i_j, \dots, i_d}$ and, moreover, the input and output buffers of their common port are occupied with the packets, where l is the limitation of the internal buffer size

(marking of places $pbl^{i_1, \dots, i_j, \dots, i_d}$, $pbl^{i_1, \dots, i_j+1, \dots, i_d}$). Such a situation constitutes the t-dead marking for the transitions of both devices while other transitions of the net $HT_{d,k}$ are potentially live.

But the structure of all the possible deadlocks is more sophisticated. We show that deadlocks occur either in cycles (chains) of blockings involving a few communicating devices (where the pair is a particular case) or because of isolation with surrounding deadlocks.

For the description of complex deadlocks of the net $HT_{d,k}$, the graph $GH_{d,k}$ of connections is constructed. In the graph $GH_{d,k}$ each node corresponds to communication device and has arcs directed to its neighbors. An example of internal node connections for $GH_{3,k}$ is shown in Fig. 4. An arc with two arrows denotes two arcs of opposite directions.

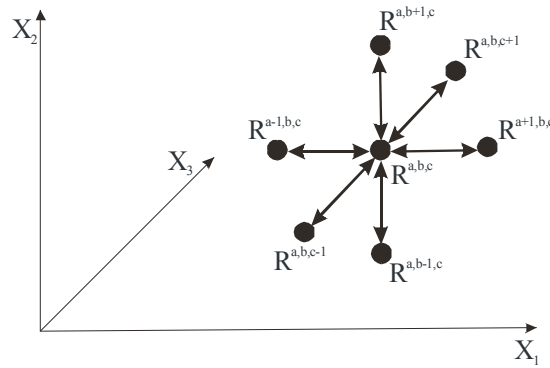


Fig. 4. Node connections of the graph $GH_{3,k}$

A directed simple cycle in the graph $GH_{d,k}$ represents a deadlock of the communication hypercube $HT_{d,k}$. In a deadlock cycle, each arc connecting a pair of neighbor devices $R^{i_1, \dots, i_u, \dots, i_d}$, $R^{i_1, \dots, i'_u, \dots, i_d}$, $|i_u - i'_u| = 1$ means that $R^{i_1, \dots, i_u, \dots, i_d}$ blocks itself iff it got l packets directed to $R^{i_1, \dots, i'_u, \dots, i_d}$, its output buffer of the port connecting $R^{i_1, \dots, i_u, \dots, i_d}$ with $R^{i_1, \dots, i'_u, \dots, i_d}$ contains a packet and the device $R^{i_1, \dots, i'_u, \dots, i_d}$ is blocked also. When the cycle ends, the last device of it blocks itself because the first device is blocked and can not receive packets.

Let us prove that all the transitions of a blocked device $R^{i_1, \dots, i_u, \dots, i_d}$ are dead. For distinctness we denote

$$r = \begin{cases} 1, & i_u - i'_u = -1, \\ 2, & i_u - i'_u = 1. \end{cases}$$

All the transitions $ti_{j,n,j',n'}^{i_1, \dots, i_d}$ are dead because marking of their input place pbl^{i_1, \dots, i_d} equals to zero, so the device cannot receive packets. All the transitions $to_{j,n}^{i_1, \dots, i_d}$, $n \neq r$ are dead because each of their input places $pb_{j,n}^{i_1, \dots, i_d}$ has zero marking. The transition $to_{u,r}^{i_1, \dots, i_d}$ is dead because marking of its input place $pol_{u,r}^{i_1, \dots, i_d}$

is zero. So the device cannot send packets. Notice that marking of $pol_{u,r}^{i_1, \dots, i_d}$ cannot be changed because $R^{i_1, \dots, i_u, \dots, i_d}$ is blocked.

Non-simple cycles of $GH_{d,k}$ represent deadlocks also. For instance, if device $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ belongs to two simple cycles and it got two output arcs directed to $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ and $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ it means that $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ blocks itself and $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$, $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ are blocked also. In this case $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ blocks itself having a_u packets directed to $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ and a_v packets directed to $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$, where $a_u + a_v = l$, and moreover each corresponding output port buffer of $R^{i_1, \dots, i_u, \dots, i_v, \dots, i_d}$ contains a packet when $a_u > 0$ ($a_v > 0$). Inductive reasoning gives the proof for d simple cycles.

The other kind of deadlocks is induced by the isolation of a device by deadlocks containing all its neighbor devices. It can be done with one simple cycle as well. For instance, in $GH_{3,k}$, the following cycle R^{i_1-1, i_2, i_3} , R^{i_1-1, i_2+1, i_3} , R^{i_1, i_2+1, i_3} , R^{i_1, i_2+1, i_3+1} , R^{i_1, i_2, i_3+1} , R^{i_1+1, i_2, i_3+1} , R^{i_1+1, i_2, i_3} , R^{i_1+1, i_2-1, i_3} , R^{i_1, i_2-1, i_3} , R^{i_1, i_2-1, i_3-1} , R^{i_1, i_2, i_3-1} , R^{i_1-1, i_2, i_3-1} , R^{i_1-1, i_2, i_3} contains all the neighbors of R^{i_1, i_2, i_3} (R^{i_1-1, i_2, i_3} , R^{i_1, i_2+1, i_3} , R^{i_1, i_2, i_3+1} , R^{i_1+1, i_2, i_3} , R^{i_1, i_2-1, i_3} , R^{i_1, i_2, i_3-1}), so the device R^{i_1, i_2, i_3} is blocked because of isolation. The isolation of a node can be generalized on the blocking of a simple chain by the isolation of its last node.

So a deadlock is a chain of blockings where the last node is blocked because:

- 1) it coincides with the first node;
- 2) it belongs to other deadlock;
- 3) it is isolated by other deadlocks.

It is very significant that occurred deadlocks create more possibilities for new deadlocks occurring. So the process has avalanche-like character. A full deadlock involving all the devices (and all the transitions) occurs when cycles (chains) contain all the devices in the hypercube. It requires at least $(l+1) \cdot k^d$ packets provided by the terminal devices. But if isolations of devices occur a little number of packets is required.

In spite of the fact that rather sophisticated hypercube communication structures were studied, the described deadlocks in the chains (cycles) of blockings and isolations are hard-nosed for real-life communication graphs, where devices with the compulsory buffering are used. We believe that these deadlocks may be purposely inflicted by the specially situated generators of the peculiar traffic. In real-life networks, the blocking of the devices is overcome with the time-out mechanisms causing the cleaning of the buffers but it leads to the considerable fall of the network performance as soon as the situation is repeated by the special generators of perilous traffic.

8 Conclusions

Thus, in the present paper, the technique of the linear invariants calculation for parametric Petri nets with the regular structure was presented. The technique was studied on the example of a communication hypercube of an arbitrary size with an arbitrary number of dimensions.

The application of the technique allowed the analysis of transmissions, involving an arbitrary number of communicating devices. The modeled telecommunication device constitutes a generalized router/switch with the compulsory buffering of the packets. Such positive properties of the communication structure as safeness and consistency were obtained using the linear invariants of infinite Petri nets.

It was grounded that the compulsory buffering of the packets inevitably leads to the possible blockings of communicating devices. The structure of the complex deadlocks involving an arbitrary number of communicating devices caused by both the chain (cycle) of blockings and the isolation was studied.

Though, in real-life networks, the deadlocks are overcome by the cleaning of the buffers via the time-out mechanism, it leads to the considerable decrease of the network performance and moreover might be inflicted by the ill-intentioned traffic.

References

- [1] Marsan A.M., Chiola G., Fumagalli A. An Accurate Performance Model of CSMA/CD Bus LAN // *Advances in Petri Nets*, LNCS, 1987, Vol. 266, P. 146-161.
- [2] Zaitsev D.A., Zaitsev I.D. Verification of Ethernet protocols via parametric composition of Petri net // *INCOM'2006: 12th IFAC/IFIP/IFORS/IEEE/IMS Symposium Information Control Problems in Manufacturing*, May 17-19 2006, Saint-Etienne, France, p. 261-267.
- [3] Zaitsev D.A. Functional Petri Nets. Universite Paris-Dauphine, Cahier du Lamsade 224, Avril 2005, 62p (www.lamsade.dauphine.fr/cahiers.html).
- [4] Shmeleva T.R. Verification of switched Ethernet protocols using infinite Petri nets // *COMINFO'2007: Third International conference on Modern information-communication technologies*, September 24-28, 2007, Levadia, Ukraine, p. 161-163.
- [5] Murata T. Petri Nets: Properties, Analysis and Applications // *Proceedings of the IEEE*, April 1989, Vol. 77, p. 541-580.
- [6] Berthelot G., Terrat R. Petri Nets Theory for the Correctness of Protocols // *IEEE Trans. on Communications*, no. 12, 1982, vol. 30, p. 2497-2505.
- [7] Diaz M. Modelling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Model // *Computer Networks*, no 6, 1982, p. 419-441.
- [8] Berthomieu B., Ribet O.-P., Vernadat F. The tool TINA - construction of abstract state space for Petri nets and Time Petri nets // *International Journal of Production Research*, Vol. 42, no. 4, 2004 (<http://www.laas.fr/tina>).

Keynote speeches

Capacity Management views



adam@metron.co.uk

Capacity Management views:
academic & pragmatic
theory & practice
real & virtual

UKPEW08
CapMan
Views

Metron

Metron & Athene

Metron

- Focus on Capacity Planning and Performance Management
- Over 350 years of CM experience & over 300 global customers
- Founding Partner in ITIL & UKCMG over 20 years ago

Athene

- Target: VMware, UNIX (inc Linux), Windows, z/OS
- Platform: Windows
- Data collectors – OS, RDBMS, user-defined
- CDB = Athene DB with Performance DB and Custom DB
- Applications: Alarms, analysis, advice, publishing, modeling...

Consulting

- Strategic studies (ITIL®, Capacity Management gap analysis...)
- Tactical projects (Athene localization, call-off...)
- Support PACS (Server consolidation, Healthcheck...)

Training

- ITIL® Capacity Management Practitioner and in Practice
- Athene (SysAdmin, Performance, Planning ...)

UKPEW08
CapMan
Views

Metron

#3 Historical Perspective

- The 'legacy' mainframe**
 - Was IBM and "the 7 dwarves" or "the BUNCH"
 - Now just IBM z/OS on zSeries
- Transition to 'open' systems**
 - Hardware costs
 - Software costs
 - Local autonomy before outsourcing
- Evolution of 'enterprise class' open systems**
 - Sun Fire UltraSPARC E20K/E25K
 - HP Integrity Superdome
 - IBM pSeries Power5 (many models)
- Consolidation and Virtualization**
 - The wheel turns full circle.

UKPEW08
CapMan
Views

Metron

#4 SDLC – Development Life Cycle

Systems	Software	Deliverables
Feasibility	Requirements	TOR, Scope, PID, SOR
Analysis	Architecture	Functional Spec
Design	Design	SPE, System spec
Implementation	Implementation	Program spec, modules
Testing	Testing	system, α , load, trials
Maintenance	Deployment	pilot, production, retire

Steps: Review, Feedback, Prototype, Cyclic
Routes: Waterfall, Iterative, Scrum.
Priorities: Project vs Process, Panic vs Procedure

UKPEW08
CapMan
Views

Metron

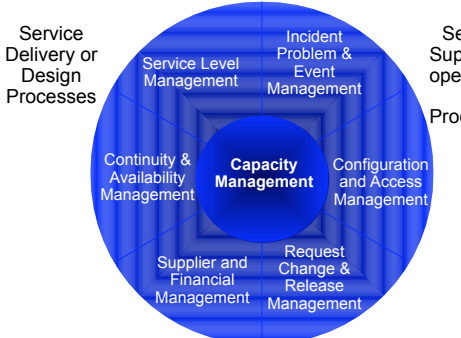
#5 ITIL® is

- IT Infrastructure Library - books & descriptions**
 - V2 exams on Service Support & Service Delivery
 - Business Perspective, Infrastructure, Development, Security & Service Management
- V3 Service Life Cycle: Strategy, Design, Transition, Operation and Improvement (and introduction)**
- Good practice for ITSM, basis of ISO/IEC 20000**
- V1 replaces GITIMM by CCTA in 80's, V2 OGC in 90's, V3 now**
- Metron key contributor to initial Demonstrator**
- itsMF, itsMFI and itsSMFUK**
 - The IT Service Management Forum for users
 - Promotes exchange of info & experience
 - UK, NL, B, AUS, ZA, F, USA (over 40 chapters).
- OGC owns it; TSO prints; itsMFI keeps; itsSMFUK actions; APMG QAs; APMG, ISEB, EXIN, Loyalist & Dansk IT examine; HP, Pink, Purple, FGI, Fox, Metron & many others teach it**
- Maybe some fragmentation with ITSM Library & ISO exams**

UKPEW08
CapMan
Views

Metron

#6 ITIL ITSM



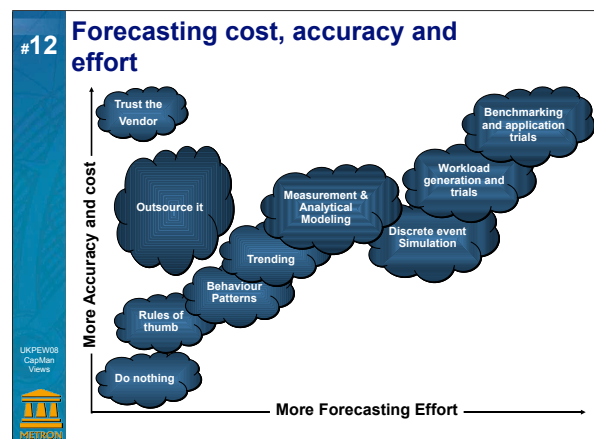
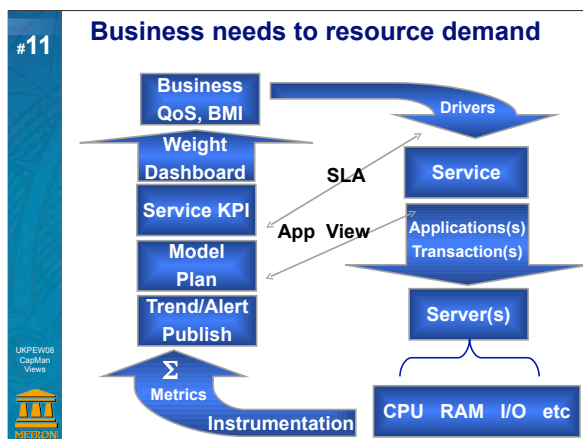
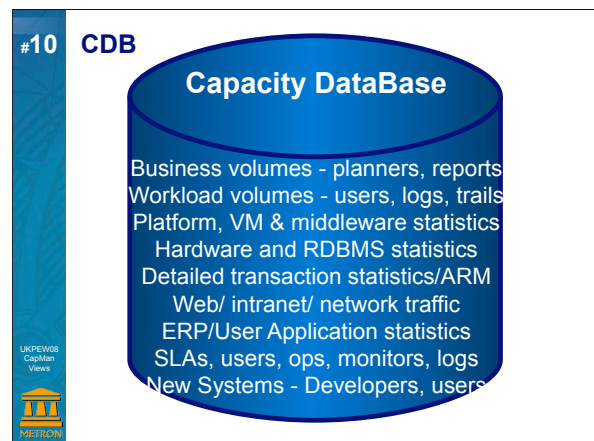
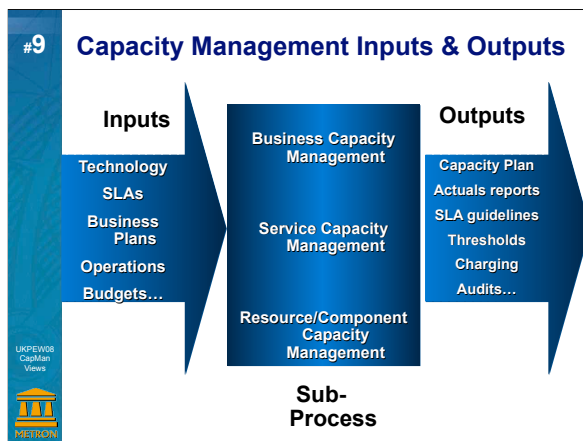
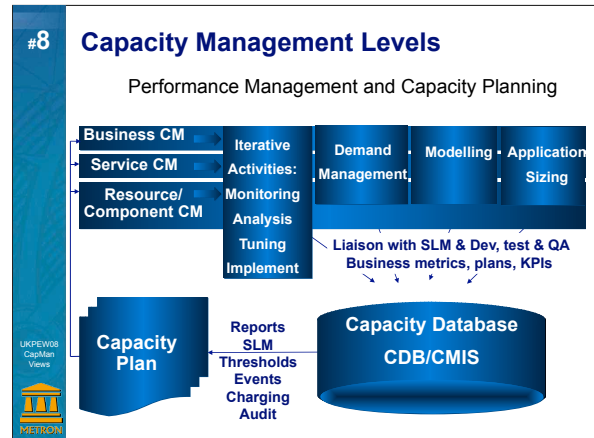
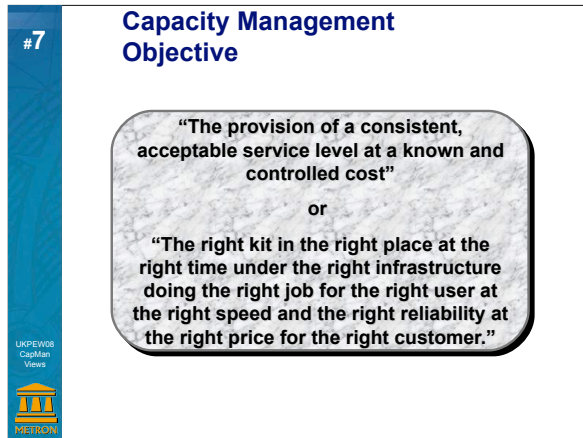
The diagram shows a central circle labeled **Capacity Management**. Surrounding it are eight ITIL processes, grouped into three categories:

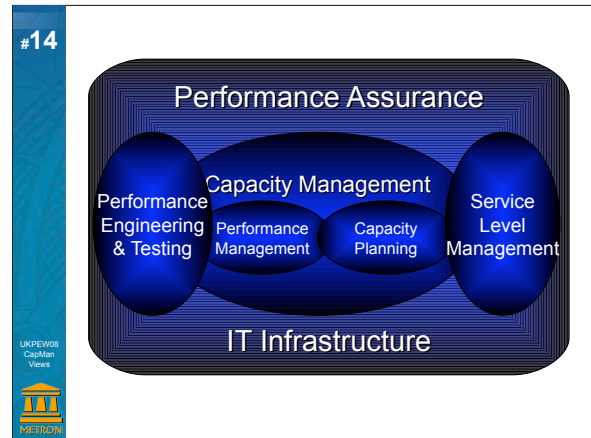
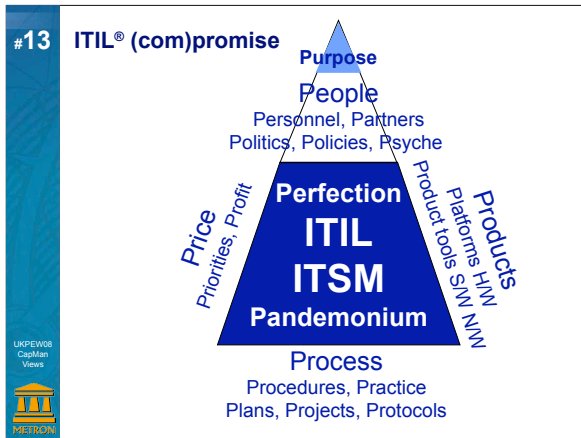
- Service Delivery or Design Processes:** Service Level Management, Continuity & Availability Management, Supplier and Financial Management.
- Incident Problem & Event Management:** Incident Problem & Event Management.
- Configuration and Access Management:** Configuration and Access Management.
- Request Change & Release Management:** Request Change & Release Management.

Service Support or operations Processes

UKPEW08
CapMan
Views

Metron

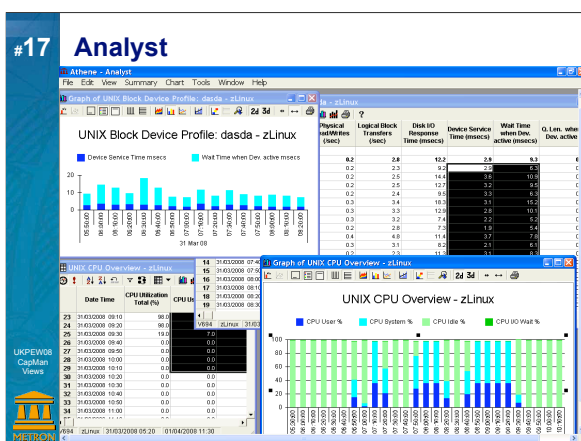
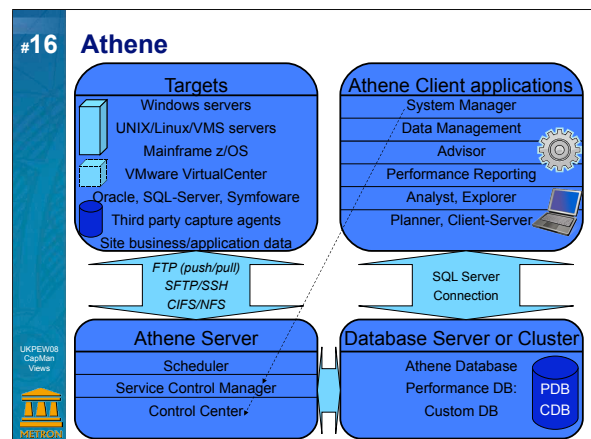


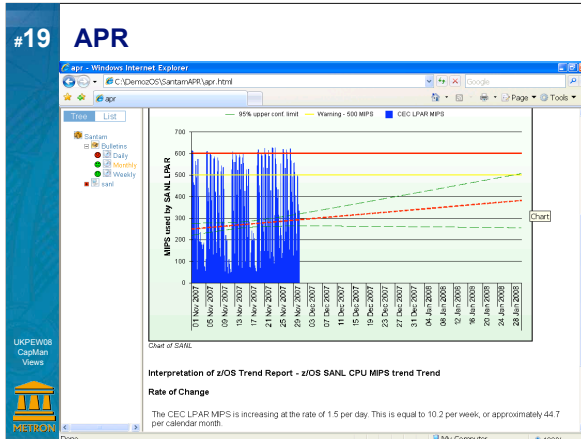


#15 CMP per app per site per stage

#	CMMI	ITSM	CapMan	Task	%
3	Optimised bITa		Business level	Dashboard, CPM	2%
4	Measured ITSM		Service level	SLAM, cap plans	10%
5	Proactive Center		Resource level	CDB, Trends, w	30%
6	Reactive Tickets		Analysis	Utilization, uptime	55%
7	Ad hoc	Help calls	Monitor	Ad hoc alerts	3%

The UKPEW08 CapMan Views logo is in the bottom left corner.





#20 Capacity Management Responsibilities

- Ensure effective resource monitoring and CDB integrity
- Ensure adequate ICT capacity
- Recommend tuning and optimization approaches for existing capacity
- Publish regular management and exception performance reports
 - Actuals versus SLA targets; Capacity issues
- Produce timely regular and ad hoc Capacity Plans
 - Maintain knowledge of future demand for IT services
 - Define need for hardware upgrades/consolidation
- Size all proposed new systems and applications
 - Performance testing of new systems/major releases
- Assess impact of new technologies at organization and ITSM levels
- Determine cost justifiable and measurable SLAs
- Recommend resolution of performance related incidents
- Recommend Demand Management strategies
- Ensure reliability and Availability requirements incorporated
- Assess Changes and advise the CAB
- Perform ad hoc self-audits.

#21 Risks for Capacity Management

- Over expectation of IT by customers
- Priority on fire-fighting/projects
- Lack of resource
- Poor external perception
- Vendor influence
 - Today's bargain is tomorrow's abacus
- Lack of information
 - Business forecasts and plans and reality
- Distributed environment
 - Need to include multi-tiers and networks.

#22 Costs of establishing the CMP

- Procurement of required tools
 - Monitoring hardware, OS, applications
 - CDB for holding record of all Capacity Management data
 - Modeling tools for "what-if" and statistics
 - Graphical reporting tools (web-enabled)
- Project management as required
- Staff costs including recruitment, training
- Accommodation etc
- Development liaison and dataflows
- QA load, soak and saturation testing data
- Business liaison and questionnaires.

#23 Benefits of Capacity Management

- Improved quality service provision and cost justifiable service quality
- Services that meet Business, Customer and User demands via integrated processes
- Increased efficiency – cost savings
 - Deferred upgrade expenditure (cash flow)
 - Consolidation (maintenance/licences)
 - Planned acquisition (discounts)
- Reduced risk
 - Fewer performance disasters (customers)
 - Fewer performance problems (users)
 - Longer Application Life (less abandoned apps)
 - Learning from previous experience with KPIs.

#24 Summary of gap analysis procedure

- Agree checklist of targets
 - Enterprise Infrastructure, Service Portfolio...
 - Mainframe, non-stop, warehouse...
 - UNIX, Linux, Windows...
 - LAN, WAN, SAN, PAN, RDBMS, ERP...
- Review the Capacity Management Process (CMP)
- Talk to the Capacity Management Team(s) (CMT)
- Interviews in a week to assess activities
- Readily available reports submitted for analysis
- Compare with checklists for Good Practice
- Review available evidence/ deliverables
- Reveal gaps (known and unknown).

10

UKPEW08
CapMan
Views



METRON

10

UKPEW08
CapMan
Views



METRON

UKPEW08
CapMan
Views



METRON

- 

1

UKPEW08
CapMan
Views



5

UKPEW08
CapMan
Views



METRON

#31 Reports required

Analysis by System per node	utilization (CPU, user, IO Wait, idle), physical utilization, Read Write requests/sec, etc
Check device by device or total per interval	
IO per interval	Physical/ Logical reads & writes per sec, % cached, etc
Free Memory per interval	CPU memory usage, paging trend/scanned sec, etc
Page swapping Summary per interval by CPU or total	Page/Swap transfer per sec, physical paging IO/sec, etc
Process per interval by CPU or total	Process switches/sec, processes running, etc
System Calls per interval	System calls/sec, System reads & writes/sec, etc
Analysis by user, user/command, command, process	
Activity per interval	Username or command/process, elapsed time, No commands, etc
Analysis per file system per interval	
	Total % inodes/blocks/KB available/used/free
RDBMS Analysis	
Overview (per interval)	CPU usage, current logons, reads/writes etc
File IO (per file id/name per interval or over period)	physical disk reads/writes, response time, Buffer cache hit ratio Logical IO summary, per session, per user etc
Session Analysis per session	Metrics, Profile, Table scans, Redo log, Latches
Correlation Analysis	Or selected combinations
All metrics vs. all metrics	
Alert Summary	of alerts, violations and severity
Alerts per interval	
Model Reports per model:	
	Response time per component (cpu/Busy/Q etc)
	Device utilization & Q per device (component)
	Memory residence & Q per component
Per scenario (per projection) -	Response Time Analysis, Throughput Analysis, Device utilization Analysis

#32 Capacity Plan template

- Management summary
 - Introduction
 - Scope of the plan and background to the server(s), service(s) involved
 - Key elements of the IT infrastructure are addressed in this plan
 - Current levels of relevant capacity in the organization
 - Problems being experienced due to under capacity
 - Degree to which service levels are being achieved
 - Outline of structure of plan, references to related documents and glossary
 - Changes since last issue of the Capacity Plan
 - Methods used to obtain information and Business data sources
 - Workload forecasts and Modeling techniques used
- Assumptions made
- Business Scenarios
- Service Summary
 - Current and recent service provision
 - Current and recent resource usage
 - Service forecasts
 - List of corresponding Workloads and Workload Forecast Scenarios
- Business Strategy
 - Service forecasts
- Options for service improvement
- Cost model
- Recommendations
 - Business benefits to be expected
 - Potential impact of carrying out the recommendations
 - Risks involved
 - Resources required
 - Costs, both set-up and ongoing

#33 SWOT with respect to the CMP

S Monitoring Event Management Project Management CMT people Previous experience	W Performance Reporting Capacity Management CDB (capacity database) CMDB (asset register) Service Processes
O Few perf metrics in PID Few perf targets in SLA Current tools Scalable tools Automated tools	T Too few staff to action Too little money to drive Too little liaison Aggregated traffic changes Business drivers & peaks

#34 Highlights

Most potential impact:

- SDLC project driver & ad hoc demands OK but add
- Matrix control with IT Infrastructure processes
 - Process workgroups to coordinate activities

Most impact already in progress:

- PID needs more performance input
 - Previous questionnaire "too long and complex"
 - Current drafts "too bland to instrument"
 - Needs more inter-team liaison to complete
- SLA needs more performance criteria
 - Business input of real requirements
 - And real traffic related to business drivers.

#35 Tools and CMP Deliverables

Domain	Tools	Used for	Deliverables
1. Enterprise	Tools	Asset Management	Monitoring
2. Mainframe	Tools	Asset Management	Performance Reporting
3. Data Center	Tools	Asset Management	Capacity Management
4. UNIX	Tools	Asset Management	Service Catalogue
5. Windows	Tools	Asset Management	SLA - availability
6. Teradata	Tools	Asset Management	SLA - availability
7. Networks	Tools	Asset Management	SLA - performance
8. Servers	Tools	Asset Management	SLA - performance
9. Storage	Tools	Asset Management	SLA - performance

#36 Actionable Items

Strategic	Tactical
Set up CDB (database)	Review metrics per domain
Set up CMIS (info system)	Review use of existing & potential tools
Add to SLA	Establish common core
Establish CMT	Review reports per domain
Outline CMP	Review use of existing & potential tools
	Establish common core
	Review top SLAs
	Add performance and capacity criteria
	Core team per domain + inter-CMT liaison
	Inter ITSM team liaison:
	monitoring, tracing, tuning,
	testing, performance engineering
	development, business
	change management & SLM.

#37

Next steps = Review and improve:

- Current Infrastructure processes and interactions
- All CMP Good Practice checklists in detail
- Real benefits of VMware program
- Pilot prototypes on top 2 services per domain:
 - CDB
 - Performance reports
 - Performance patterns and thresholds
 - Performance analyses and trends
 - Capacity plans
- Pilot prototypes on top 2 servers per domain
 - Workload characterization
 - Workload forecast scenarios
 - Resultant resource demand scenarios.



#38

CMMI “now” and “then”

	CMMI Now	CMMI Then	Comments, notes
1. ITSM Capacity Management is in place	●	●	Only CMF is in operation area
1. CM is mandatory	●	●	Essentially ad-hoc or reactive only
2. CM determines agreed services for action	●	●	Services mostly well known, but not tagged
3. Capacity Plans are produced	●	●	Plans created by teams to ensure business
4. A CDB is maintained	●	●	No formal history or CDB across platforms
5. Future capacity demand is forecasted	●	●	% increase for major new projects
2. Capacity Management is active	●	●	
1. CM activities are in place	●	●	Reactive for most significant services
2. CM sub-processes are in place	●	●	Only resources but not only for some services
3. CM team has documented processes	●	●	No specific team as such in most domains
3. CM inputs are in place	●	●	Tools in place but not fully exploited
1. Tools are in place for agreed metrics	●	●	Starting to ask for more info in PIC
2. Agreed data is provided by business units	●	●	
4. CM Reports are in place	●	●	
1. Resource utilization	●	●	Reviewed within teams
2. Performance and workload trends	●	●	Monthly/quarterly % of new projects
3. Details of proposed new workloads	●	●	PIC knows to which info, but no notification
4. Recommendations	●	●	Based on data, not on metrics
5. Variations	●	●	For objective performance criteria set
6. Key SLA performance is achieved	●	●	No to have this performance criteria
7. Capacity plans issued for agreed services	●	●	Capacity conclusions generated within teams
4. CM interfaces with ITSM (see 11.2)	●	●	
5. Good IT Practice is being observed	●	●	
1. Standards including standard builds etc	●	●	
2. Responsibility inc. notification of changes	●	●	
3. Suitable tools	●	●	Tools in place but not fully exploited
6. User Perception is measured	●	●	
7. Feedback is collected, analysed & acted on	●	●	But user feedback has with app dev.



Capacity Management views





*Capacity Management views:
academic & pragmatic
theory & practice
real & virtual*



Large-Scale Parallel Computing on Grids

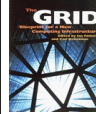
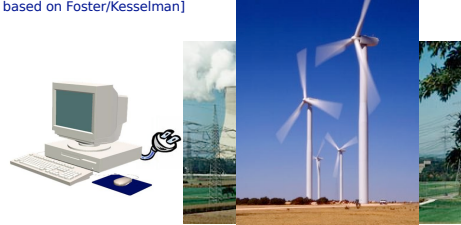
Henri Bal
bal@cs.vu.nl
Vrije Universiteit Amsterdam

7th Int. Workshop on
Methodologies and
Tools for
29


The 'Promise of the Grid'

Efficient and transparent (i.e. easy-to-use) wall-socket computing over a distributed set of resources [Sunderam ICCS'2004, based on Foster/Kesselman]

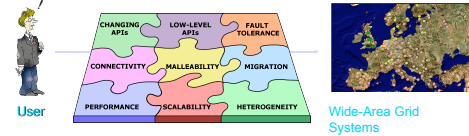
Parallel computing on grids

- Mostly limited to
 - trivially parallel applications
 - parameter sweeps, SETI@home, RSA-155
 - applications that run on one cluster at a time
 - use grid to schedule application on a suitable cluster
- Our goal: run real parallel applications on a large co-allocated resources




Reality: 'Problems of the Grid'

- Performance & scalability
- Heterogeneous
- Low-level & changing programming interfaces
- Connectivity issues
- Fault tolerance
- Malleability




• writing & deploying grid applications is hard

It's a jungle out there



Some solutions

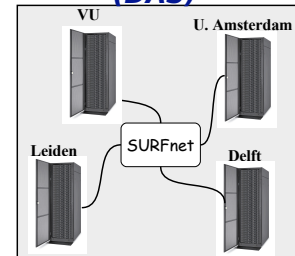
- Performance & scalability
 - Fast optical private networks (OPNs)
 - Algorithmic optimizations
- Heterogeneity
 - Use Virtual Machine technology (e.g. Java VM)
- Grid programming systems dealing with connectivity, fault tolerance ...
 - Ibis programming system
 - GAT (Grid Application Toolkit)



Outline

- Grid infrastructures
 - DAS: a Computer Science grid with a 40 Gb/s optical wide-area network
- Parallel algorithms and applications on grids
 - Search algorithms, Awari
 - DiViNE model checker (early experiences)
- Programming environments
 - Ibis and JavaGAT

Distributed ASCI Supercomputer (DAS)



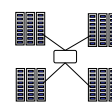
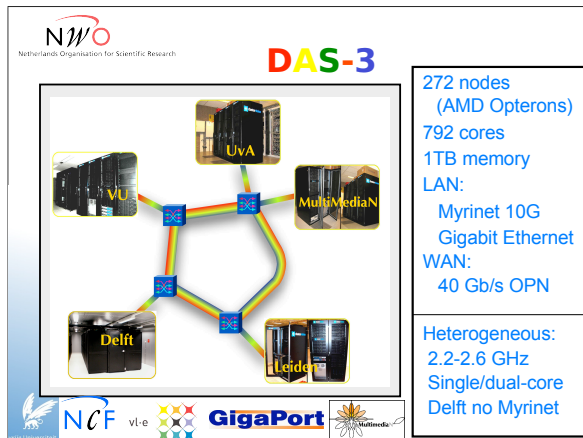
DAS-1 (1997)



DAS-2 (2002)

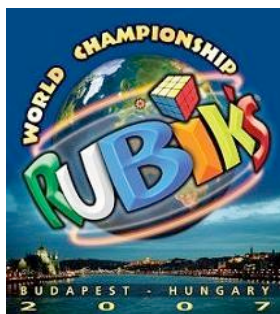


DAS-3

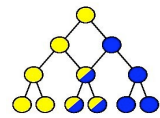


Wide-area algorithms

- Key problem is fighting *high latency*
 - LAN: 1-100 usec, WAN: 1-100 msec
- Experience Albatros project: traditional optimizations can be very effective for grids
 - Use asynchronous communication
 - Allows latency hiding & message combining
 - Exploit hierarchical structure of grids
 - E.g. load balancing, group communication

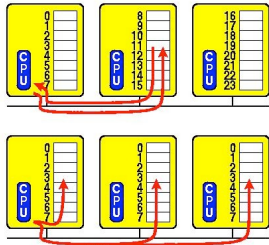


Example 1: heuristic search



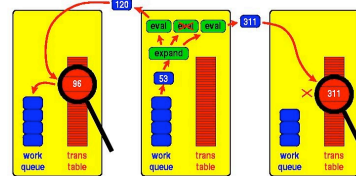
- Parallel search: partition search space
- Many applications have *transpositions*
 - Reach same position through different moves
- Transposition table:
 - Hash table storing positions that have been evaluated before
 - Shared among all processors

Distributed transposition tables



- Partitioned tables
- 10,000s synchronous messages per second
- Poor performance even on a cluster
- Replicated tables
- Broadcasting doesn't scale (many updates)

Transposition Driven Scheduling

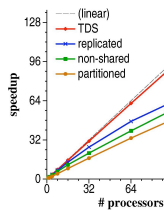


- Send job asynchronously to owner table entry
- Can be overlapped with computation
- Random (=good) load balancing
- Delayed & combined into fewer large messages

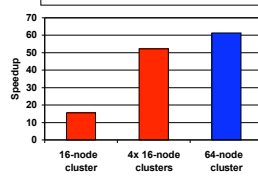


Speedups for Rubik's cube

Single Myrinet cluster



TDS on wide-area DAS-2



- Latency-insensitive algorithm works well even on a grid
- Despite huge amount of communication

Example 2: Solving awari



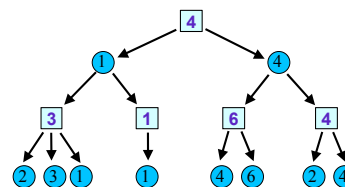
- Solved by John Romein [IEEE Computer, Oct. 2003]
- Computed on a Myrinet cluster (DAS-2)
- Recently used wide-area DAS-3 [CCGrid, May 2008]
- Determined score for **889,063,398,406** positions
- Game is "draw"
Andy Tanenbaum: "You just ruined a perfectly fine 3500 year old game"

Rules of awari



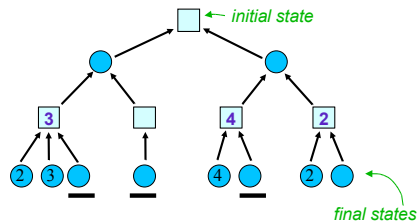
- Start with $4 \times 12 = 48$ stones
- Sow counterclockwise
- Capture if last, enemy pit contains 2-3 stones
- Goal: capture majority of stones

(We don't use) MiniMax



- Depth-first search (e.g., alpha-beta)

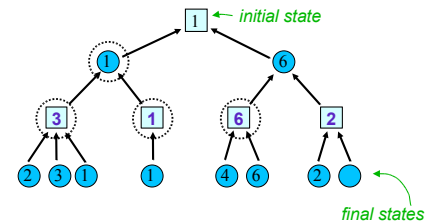
Retrograde analysis



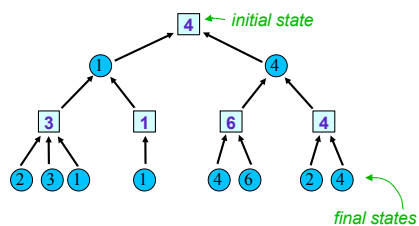
- Start from final positions with known outcome
- Like mate, stalemate in chess

Propagate information upwards

Retrograde analysis



Retrograde analysis

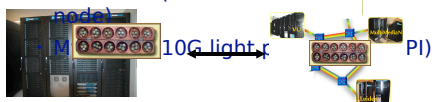


Parallel retrograde analysis

- Partition database, like transposition tables
- Random distribution (hashing), good load balance
- Repeatedly send results to parent nodes
 - Asynchronous, combined into bulk transfers
- Extremely communication intensive:
 - 1 Pbit of data in 51 hours
 - What happens on a grid?

Awari on DAS-3 grid

- Implementation on single big cluster
 - 144 cores (1 cluster of 72 nodes, 2 cores/node)
 - Myrinet (MPI)
- Naïve implementation on 3 small clusters
 - 144 cores (3 clusters of 24 nodes, 2 cores/node)



Initial insights

- Single-cluster version has high performance, despite high communication rate
 - Up to 28 Gb/s cumulative network throughput
- Naïve grid version has flow control problems
 - Faster CPUs overwhelm slower CPUs with work
 - Unrestricted job queue growth
 - => Add regular global synchronizations (barriers)
- Naïve grid version 2.1x slower than 1 big

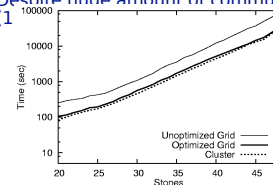
Optimizations

- Scalable barrier synchronization algorithm
 - Ring algorithm has too much latency on a cluster
 - Tree algorithm for barrier & termination detection
 - Better flushing strategy while in termination phase
- Reduce host overhead
 - CPU overhead for MPI message handling/polling
 - Don't just look at latency and throughput
- Optimize grain size per network (LAN vs. WAN)
 - Large messages (much combining) have lower host overhead but higher load-imbalance



Performance

- Optimizations improved grid performance by 50%
- Grid version only 15% slower than 1 big cluster
- Despite huge amount of communication (100,000 messages per database)



DiVinE

- Developed at Masaryk University Brno (ParaDiSe Labs, Jiri Barnat – main architect)
- Distributed Verification Environment
 - Tool for distributed state space analysis and LTL model-checking
 - Development and fast prototyping environment
- Ported to DAS-3 by Kees Verstoep

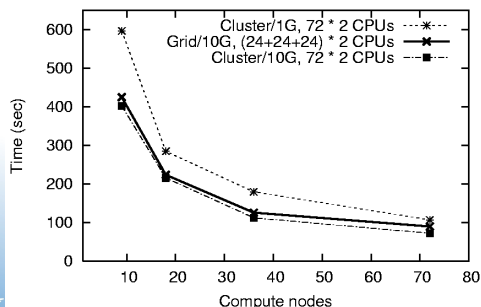


DiVinE experiments on DAS-3

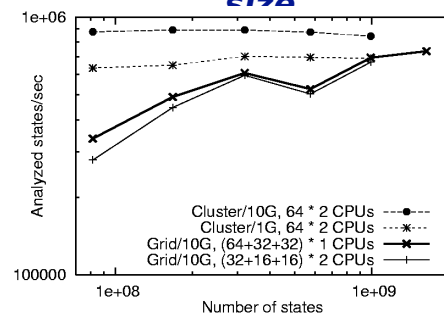
- Tried difficult problems from BEEM database
 - Maximal Accepting Predecessor version
- Experiments:
 - 1 cluster, 10 Gb/s Myrinet
 - 1 cluster, 1 Gb/s Ethernet
 - 3 clusters, 10 Gb/s light paths
- Up to 144 cores (72 nodes) in total



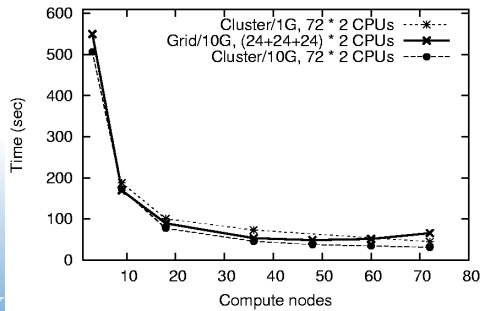
Elevator-1



Elevator-1: scaling problem size



Anderson



Preliminary insights

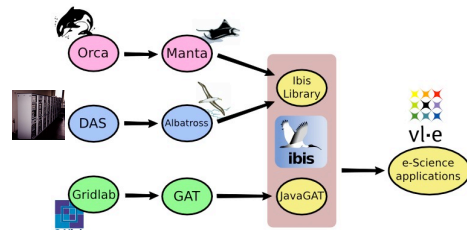
- Many similarities between Awari and DiVinE
 - Similar communication patterns, data rates, random state distribution
- Largest problem solved so far:
 - 1.6 billion states on 128 (64+32+32) nodes
 - 350 GB memory, 40 minutes WAN capacity close to 10G
- We need bigger problems
 - Industry-strength challenges

Outline

- Grid infrastructures
 - DAS: a Computer Science grid with a 40 Gb/s optical wide-area network
- Parallel algorithms and applications on grids
 - Search algorithms, Awari
 - DiVinE model checker (early experiences)
- Programming environment
 - Ibis and JavaGAT



Research at the VU



The Ibis Project

- Goal:
 - drastically simplify grid programming/deployment
- Used for many large-scale grid experiments
 - Non-trivially parallel applications running on many heterogeneous resources (co-allocation)



Approach (1)

- Write & go: minimal assumptions about execution environment
 - Virtual Machines (Java) deal with heterogeneity
- Use middleware-independent APIs
 - Mapped automatically onto middleware
- Different programming abstractions
 - Low-level message passing
 - High-level divide-and-conquer

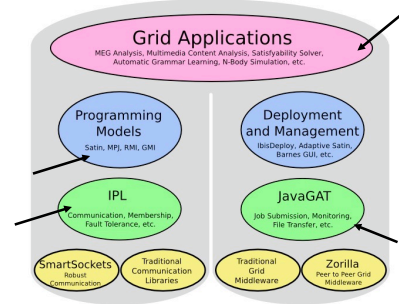


Approach (2)

- Designed to run in dynamic/hostile grid environment
 - Handle fault-tolerance and malleability
 - Solve connectivity problems automatically (SmartSockets)
- Modular and flexible: can replace Ibis components by external ones
 - Scheduling: Zorilla P2P system or external broker



Global picture



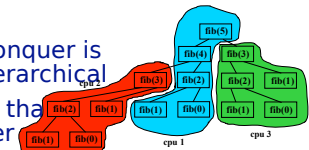
Ibis Portability Layer (IPL)

- A “run-anywhere” communication library
 - Sent along with the application (jar-files)
- Support malleability & fault-tolerance
 - Change number of machines during the run
- Efficient communication
 - Configured at startup, based on capabilities (multicast, ordering, reliability, callbacks)

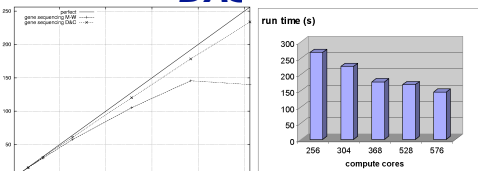


Satin: divide-and-conquer

- Divide-and-conquer is inherently hierarchical
- More general than master/worker
- Cilk-like primitives (spawn/sync) in Java
- Fault tolerance: can crash and recover



Gene sequence comparison in Satin (on DAS-2)



Speedup on 1 cluster, up to 256 cores

Run times on 5 clusters, up to 576 cores

- Divide&conquer scales much better than master-worker



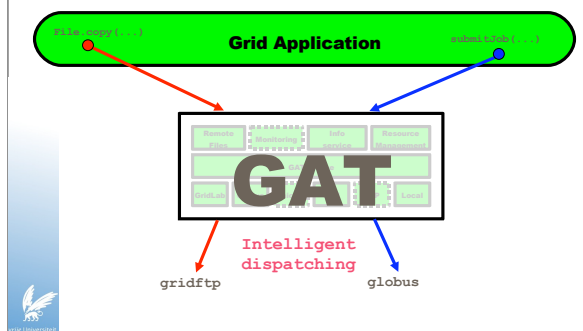
JavaGAT



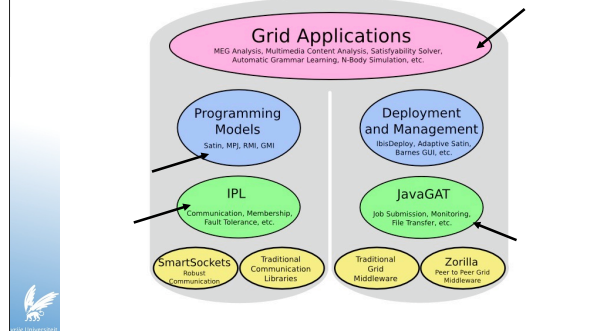
- GAT: Grid Application Toolkit
 - Makes grid applications independent of the underlying grid infrastructure
- Used by applications to access grid services
 - File copying, resource discovery, job submission & monitoring, user authentication
- API is currently standardized (SAGA)
 - SAGA being implemented on JavaGAT



Grid Applications with GAT



Global picture



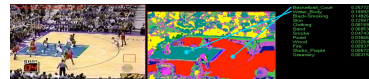
Ibis applications

- Multimedia content analysis
- SAT-solver
- Brain imaging
- Mass spectroscopy
- N-body simulations
-
- Other programming systems
 - Workflow engine for astronomy (D-grid), grid file system, ProActive, Jylab, ...

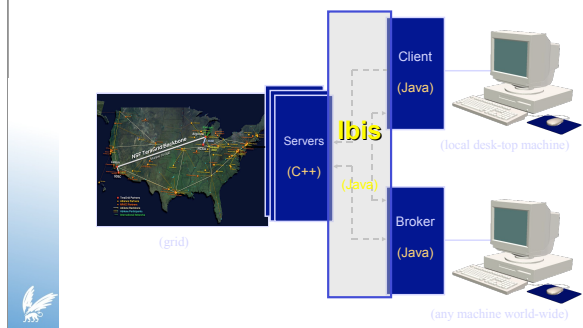


Multimedia content analysis

- Analyzes video streams to recognize objects
- Extract feature vectors from images
 - Describe properties (color, shape)
 - Data-parallel task implemented with C++/MPI
- Compute on consecutive images



MMCA application

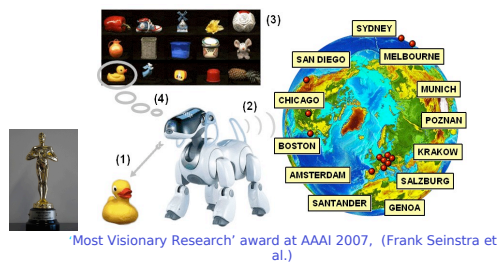


MMCA with Ibis

- Initial implementation with TCP was unstable
- Ibis simplifies communication, fault tolerance
- SmartSockets solves connection problems
- Clickable deployment interface
- Demonstrated at many conferences (SC'07)
- 20 clusters on 3 continents, 500-800



MMCA



Summary

- Goal:
 - Parallel applications on large-scale grids
- Need efficient distributed algorithms for grids
 - Latency-tolerant (asynchronous)
 - Available bandwidth is increasing
- Early experiments with DiVinE are promising
- High-level grid programming

Acknowledgements

Current members Past members

Rob van Nieuwpoort
 Jason Maassen
 Thilo Kielmann
 Frank Seinstra
 Niels Drost
 Cerial Jacobs
 Kees Verstoep
 Roelof Kemp
 Kees van Reeuwijk

John Romein
 Gosia Wrzesinska
 Rutger Hofman
 Maik Nijhuis
 Olivier Aumage
 Fabrice Huet
 Alexandre Denis

More information

- Ibis can be downloaded from
 - <http://www.cs.vu.nl/ibis>
- Ibis tutorials
 - Next one at CCGrid 2008 (19 May, Lyon)

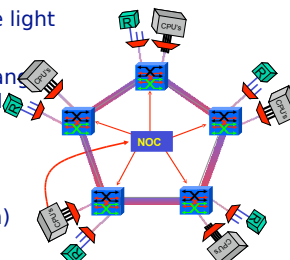
CCGrid2008

EXTRA

GigaPort

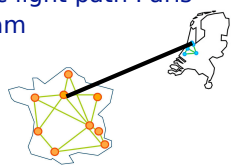
StarPlane

- Applications can dynamically allocate light paths
- Applications can change topology of the WAN
- Joint work with Cees de Laat (Univ. of Amsterdam)



DAS-3 and Grid'5000

- 10 Gb/sec light path Paris - Amsterdam



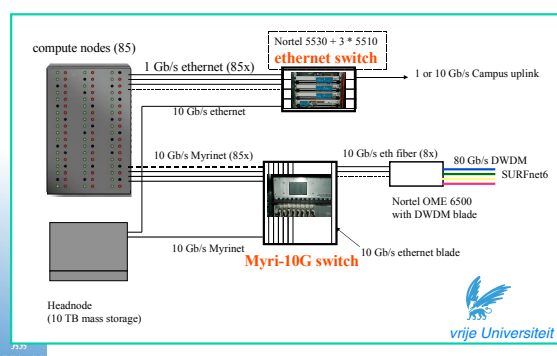
- First step towards a European Computer Science grid?
- See Cappello's keynote @



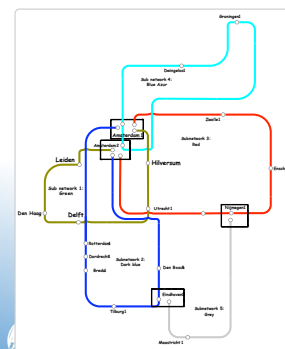
Configuration

	LU	TUD	UvA-VLe	UvA-MN	VU	TOTALS
Head						
* storage	10TB	5TB	2TB	2TB	10TB	29TB
* CPU	2x2.4GHz DC	2x2.4GHz DC	2x2.2GHz DC	2x2.2GHz DC	2x2.4GHz DC	
* memory	16GB	16GB	8GB	16GB	8GB	64GB
* Myri 10G	1		1	1	1	
* 10GE	1	1	1	1	1	
Compute						
* storage	400GB	250GB	250GB	2x250GB	250GB	84 TB
* CPU	2x2.6GHz	2x2.4GHz	2x2.2GHz DC	2x2.4GHz	2x2.4GHz DC	1.9 THz
* memory	4GB	4GB	4GB	4GB	4GB	1048 GB
* Myri 10G	1		1	1	1	
Myrinet						
* 10G ports	33 (7)		41	47	86 (2)	
* 10GE ports	8		8	8	8	320 Gb/s
Nortel						
* 1GE ports	32 (16)	136 (8)	40 (8)	46 (2)	85 (11)	339 Gb/s
* 10GE ports	1 (1)	9 (3)	2	2	1 (1)	

DAS-3 networks



Current SURFnet6 network

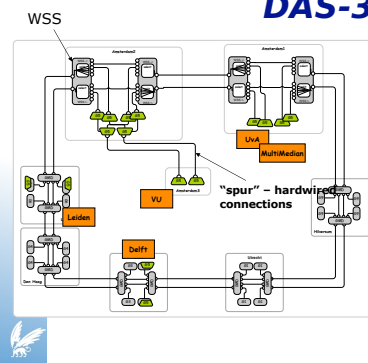


- 4 DAS-3 sites happen to be on Subnetwork 1 (olive green)

- Only VU needs extra connection

Source: Erik-Jan Bos (Siren'06 talk)

Adding 10G waves for DAS-3



- Band 5 (up to 8 channels) added at all participating nodes
- Wavelength Selective Switches (WSS) added to support reconfigurability
- "Spur" to connect VU
- Full photonic mesh now possible between DAS-3 sites

Sequential Fibonacci

```
public long fib(int n) {
    if (n < 2) return n;

    long x = fib(n - 1);
    long y = fib(n - 2);

    return x + y;
}
```

Parallel Fibonacci

```
interface FibInterface extends ibis.satin.Spawnable {
    public long fib(int n);
}

public long fib(int n) {
    if (n < 2) return n;

    long x = fib(n - 1);
    long y = fib(n - 2);
    sync();
    return x + y;
}
```



Parallel Fibonacci

```
interface FibInterface extends ibis.satin.Spawnable {
    public long fib(int n);
}

public long fib(int n) {
    if (n < 2) return n;

    long x = fib(n - 1);
    long y = fib(n - 2);
    sync();
    return x + y;
}
```

Mark methods as
Spawnable.
They can run in parallel.



Parallel Fibonacci

```
interface FibInterface extends ibis.satin.Spawnable {
    public long fib(int n);
}

public long fib(int n) {
    if (n < 2) return n;

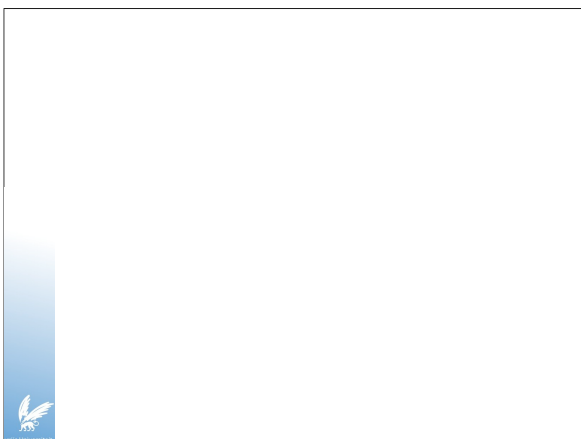
    long x = fib(n - 1);
    long y = fib(n - 2);
    sync();
    return x + y;
}
```

Mark methods as
Spawnable.
They can run in parallel.

Wait until spawned
methods are done.



Elevator-1 times



Index

- Abdulai, J., 300
Awan, I.U., 172, 217
Azgomi, M.A., 107

Bal, H., 379
Bidgoly, A.J., 107
Bodrog, L., 89

Chen, Y., 226
Chester, A.P., 199
Clark, A., 64
Clegg, R.G., 48
Cohen, J., 38

Darlington, J., 38
Dingle, N.J., 246, 347

Etorban, A., 271

Fan, L., 285

Gilmore, S., 64
Griffin, D., 48
Grummitt, A., 372

Hammond, S.D., 1
Harrison, P.G., 262
Haverkort, B.R., 76
He, L., 199
Holton, D.R.W., 217
Horváth, G., 89
al-Humoud, S.O., 300

Ibrahim, I.S., 271
Ioualalen, M., 123

Jarvis, S.A., 1, 199
Jongerden, M.R., 76

Kassab, H.Y., 137
Khan, A.K., 311
King, P.J.B., 271
Knottenbelt, W.J., 246, 347

Landa, R., 48
Lebrecht, A.S., 246

Mackenzie, L.M., 300
Martínez Ortuño, F., 337
Martini, P., 148
van Moorsel, A., 137
Moreaux, P., 123
Mudalige, G.R., 1
Mykoniati, E., 48

Osman, R., 172
Ould-Khaoua, M., 300

Pustina, L., 148

Reinecke, P., 18
Richardson, C., 38
Rio, M., 48

Salmi, N., 123
Schwarzer, S., 148
Shang, D., 226
Shmeleva, T.R., 358
Smith, J.A., 1
Sun, Z., 237, 285

Thomas, N., 27, 160
Thornley, D.J., 184, 326

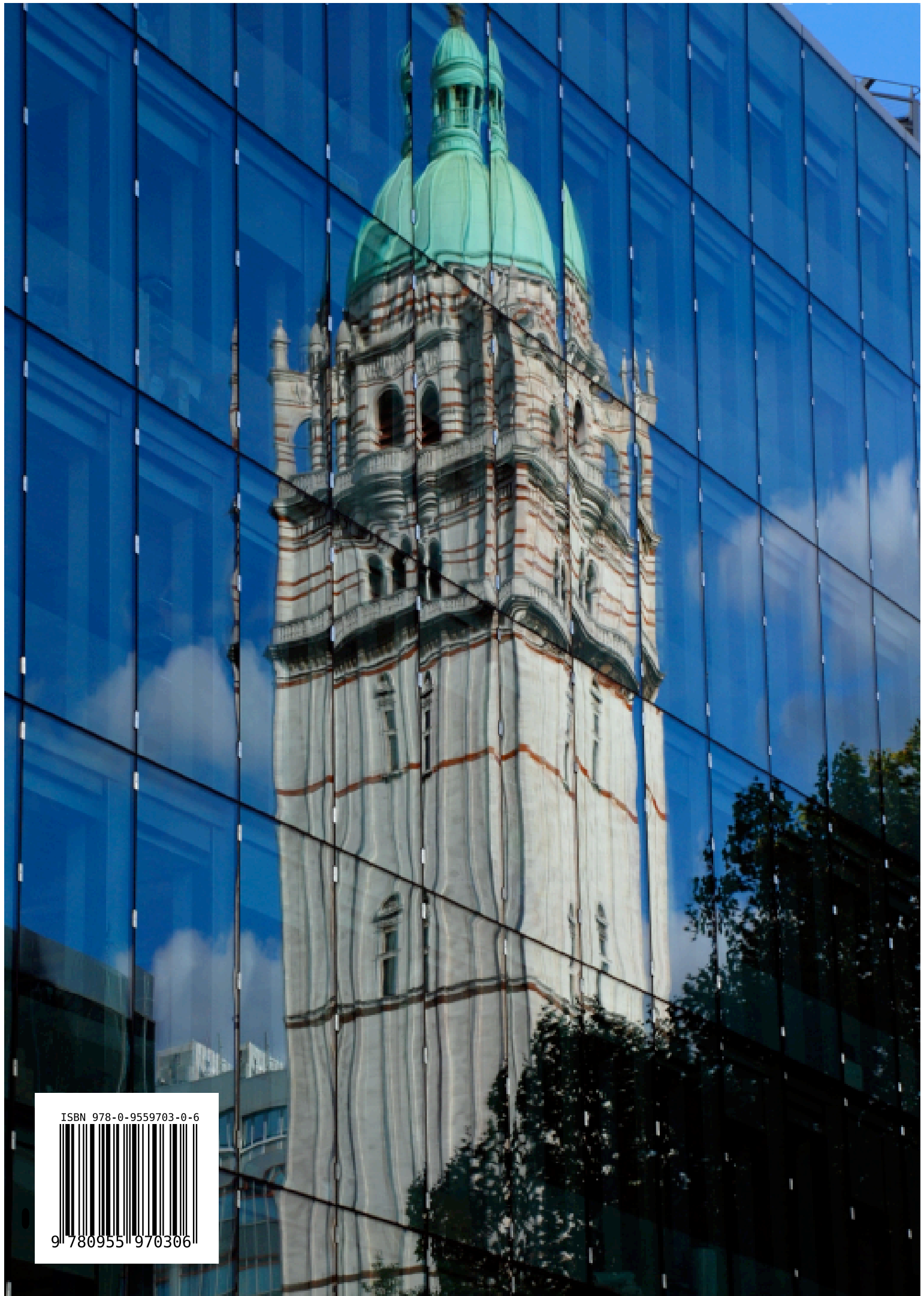
Vigliotti, M.G., 262
Vulkán, C., 89

Wolter, K., 18
Woodward, M.E., 172
Wu, J., 237, 285

Xia, F., 226
Xue, J.W.J., 199

Yakovlev, A., 226
Yang, Y., 285
Younas, M., 217

Zaitsev, D.A., 358
Zhao, Y., 160
Zhou, Y., 285



ISBN 978-0-9559703-0-6



9 780955 970306