Response Time Densities in Generalised Stochastic Petri Net Models

Nicholas J. Dingle, Peter G. Harrison and William J. Knottenbelt
Department of Computing, Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, United Kingdom
{njd200,pgh,wjk}@doc.ic.ac.uk

ABSTRACT

Generalised Stochastic Petri nets (GSPNs) have been widely used to analyse the performance of hardware and software systems. This paper presents a novel technique for the numerical determination of response time densities in GSPN models. The technique places no structural restrictions on the models that can be analysed, and allows for the high-level specification of multiple source and destination markings, including any combination of tangible and vanishing markings. The technique is implemented using a scalable parallel Laplace transform inverter that employs a modified Laguerre inversion technique. We present numerical results, including a study of the full distribution of end-to-end response time in a GSPN model of the Courier communication protocol software. The numerical results are validated against simulation.

1. INTRODUCTION

Generalised Stochastic Petri nets (GSPNs) [3] are a popular graphical modelling formalism for investigating the qualitative and quantitative properties of complex concurrent systems. Hardware and software systems that have been modelled successfully with GSPNs include communication protocols, parallel programs, multiprocessor memory caches and distributed databases [15, 16].

The main objective in the quantitative analysis of GSPNs to date has been to obtain the equilibrium probability distribution of their underlying markings, particularly for models with large reachability graphs (e.g. [6, 7, 5, 14]). From this, standard resource-based measures can be derived, for example mean buffer occupancies, system availablility and throughput. *Expected* values of various sojourn times (corresponding to the mean time taken for the net to evolve from a given source marking to a given destination marking in any number of transition firings) can also be obtained in this way.

The focus of the present study, however, is on the harder problem of calculating the full distribution of such response times. We take as input a GSPN model, a time range and a high-level description of the source and destination markings, and give as output a graph of the corresponding response time density. We achieve this in two main steps. First, we construct a system of linear equations which can be solved to yield the value of the Laplace transform of the response time density for arbitrary values of the (complex) transform parameter s. By working in the Laplace domain we are able to exploit the property that the Laplace transform of the density of a sum of independent random variables is the product of the Laplace transforms of the densities of the individual random variables. We then derive the response time density curve from its Laplace transform by using a scalable parallel numerical transform inverter based on a modified Laguerre method.

The rest of this paper is organised as follows. Section 2 introduces definitions of relevant terminology. Section 3 derives the linear systems which are solved to yield the Laplace transform of the density of the response time between the specified source and destination markings. Both single and multiple source markings are considered, the latter problem involving the assignment of appropriate weights to each source marking to determine an unconditional Laplace transform. Inversion of the Laplace transform to obtain the response time density is considered in Section 4. Section 5 details a complete distributed response time analysis pipeline, the heart of which is a highly scalable distributed transform inverter. Section 6 presents results for two case studies, namely a small contrived GSPN and a complex GSPN model of a communication protocol. Section 7 concludes and considers future work.

2. PRELIMINARIES

Generalised Stochastic Petri nets are extensions of Place-Transition nets, which are untimed Petri nets with no transition firing delays. A Place-Transition net is formally defined as [4]:

DEFINITION 1.
A Place-Transition net is a 5-tuple $PN = (P, T, I^-, I^+, M_0)$ *where:*

- $P = \{p_1, \ldots, p_n\}$ *is a finite and non-empty set of places.*
- $T = \{t_1, \ldots, t_m\}$ *is a finite and non-empty set of transitions.*
- $P \cap T = \emptyset$.
- $I^-, I^+ : P \times T \rightarrow \mathbb{N}_0$ *are the backward and forward incidence functions, respectively. If* $I^-(p,t) > 0$, *an arc leads from place* p *to transition* t, *and if* $I^+(p,t) > 0$ *then an arc leads from transition* t *to place* p.
- $M_0 : P \rightarrow \mathbb{N}_0$ *is the initial marking defining the initial number of tokens on every place.*

A marking is a vector of integers representing the number of tokens on each place in a Petri net. The set of all markings that are reachable from the initial marking M_0 is known as the *state space* or *reachability set* of the Petri net, and is denoted by $R(M_0)$. The connections between markings in the reachability set form the *reachability graph*. Formally, if the firing of a transition that is enabled in marking M_i results in marking M_j, then the reachability graph contains a directed arc from marking M_i to marking M_j.

GSPNs [3] are timed extensions of Place-Transition nets with two types of transitions: *immediate* transitions and *timed* transitions. Once enabled, immediate transitions fire in zero time, while timed transitions fire after an exponentially distributed firing delay. Firing of immediate transitions therefore has priority over the firing of timed transitions.

The formal definition of a GSPN is as follows [4]:

DEFINITION 2.
A GSPN is a 4-tuple $GSPN = (PN, T_1, T_2, W)$ where:

- $PN = (P, T, I^-, I^+, M_0)$ *is the underlying Place-Transition net.*
- $T_1 \subseteq T$ *is the set of timed transitions,* $T_1 \neq \emptyset$,
- $T_2 \subset T$ *denotes the set of immediate transitions,* $T_1 \cap T_2 = \emptyset$, $T = T_1 \cup T_2$
- $W = (w_1, \ldots, w_{|T|})$ *is an array whose entry* w_i *is either*
  - *a (possibly marking dependent)* **rate** $\in \mathbb{R}^+$ *of an exponential distribution specifying the firing delay, when transition* t_i *is a timed transition, i.e.* $t_i \in T_1$ *or*
  - *a (possibly marking dependent)* **weight** $\in \mathbb{R}^+$ *specifying the relative firing frequency, when transition* t_i *is an immediate transition, i.e.* $t_i \in T_2$.

The reachability graph of a GSPN contains two types of markings. A vanishing marking is one in which an immediate transition is enabled. The sojourn time in such markings is zero. A tangible marking is one which enables only timed transitions. The sojourn time in such markings is exponentially distributed. We denote the set of reachable vanishing markings by $\mathcal{V}$ and the set of reachable tangible markings by $\mathcal{T}$.

We define p_{ij} to be the probability that j is the next marking entered after marking i, μ_i^{-1} to be the mean sojourn time in marking i and, for $i \in \mathcal{T}$, $q_{ij} = \mu_i p_{ij}$; i.e. q_{ij} is the instantaneous transition rate into marking j from marking i.

3. RESPONSE TIME EQUATIONS

The first passage time from a single source marking i into a non-empty set of destination markings $\vec{j}$ is:

$$T_{i\vec{j}} = \inf\{t > 0 : \; M(t) \in \vec{j} \mid M(0) = i\}$$

where $M(t)$ is the marking of the Petri net at time t.

Let $f_{i\vec{j}}(t)$ be the probability density function of $T_{i\vec{j}}$ and $L_{i\vec{j}}(s)$ be its Laplace transform, i.e.

$$L_{i\vec{j}}(s) = \int_0^\infty e^{-st} f_{i\vec{j}}(t)dt$$

for complex s with real part $\mathrm{Re}(s) > 0$. We compute this Laplace transform using the first-step analysis approach described in the next section, and invert it numerically using the modified Laguerre method described in Section 4.

3.1 Derivation for a single source marking

The Laplace transform of the (exponential) sojourn time density function of tangible marking i is $\mu_i/(s + \mu_i)$. For a vanishing marking, sojourn time is zero with probability 1, giving a corresponding Laplace transform of 1 for all s.

Since state holding times are independent and the Markov property holds at transition firing instants, a first step analysis gives:

$$L_{i\vec{j}}(s) = \begin{cases} \sum_{k \notin \vec{j}} \left(\frac{q_{ik}}{s+\mu_i}\right) L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} \left(\frac{q_{ik}}{s+\mu_i}\right) & \text{if } i \in \mathcal{T} \\ \sum_{k \notin \vec{j}} p_{ik} L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} p_{ik} & \text{if } i \in \mathcal{V} \end{cases}$$

For example, when $\vec{j} = \{1\}$, $\mathcal{V} = \{2\}$ and $\mathcal{T} = \{1, 3, 4, 5, \ldots, n\}$ the above equations can be written as:

$$\begin{pmatrix} s+\mu_1 & -q_{12} & \cdots & -q_{1n} \\ 0 & 1 & \cdots & -p_{2n} \\ 0 & -q_{32} & \cdots & -q_{3n} \\ 0 & \vdots & \ddots & \vdots \\ 0 & -q_{n2} & \cdots & s+\mu_n \end{pmatrix} \begin{pmatrix} L_{1\vec{j}}(s) \\ L_{2\vec{j}}(s) \\ L_{3\vec{j}}(s) \\ \vdots \\ L_{n\vec{j}}(s) \end{pmatrix} = \begin{pmatrix} 0 \\ p_{21} \\ q_{31} \\ \vdots \\ q_{n1} \end{pmatrix} \quad (1)$$

In practice, we have found that numerical problems can arise when solving these equations if there is a large difference in the magnitude of s relative to that of the rates μ_i. This is due to the limitations of finite precision floating point arithmetic. However, potential problems can easily be detected before solving these equations and can often be resolved by scaling all transition rates in the model by a constant factor. In pathological cases where the fastest and slowest rates in the model differ by several orders of magnitude, the faster rates can typically be replaced by immediate transitions to a good approximation.

3.2 Multiple source markings

When there are multiple source markings, denoted by the vector $\vec{i}$, the Laplace transform of the response time density at equilibrium is:

$$L_{\vec{i}\vec{j}}(s) = \sum_{k \in \vec{i}} \alpha_k L_{k\vec{j}}(s)$$

where the weight α_k is the equilibrium probability that the marking is $k \in \vec{i}$ at the starting instant of the passage. This instant is the moment of entry into marking k; thus α_k is proportional to the equilibrium probability of the marking k in the underlying embedded (discrete-time) Markov chain (EMC) defined by the marking of the GSPN at firing instants. This EMC is characterised by the one-step transition probability matrix P with elements p_{ij} defined in Section 2. Therefore,

$$\alpha_k = \frac{\tilde{\pi}_k}{\sum_{j \in \vec{i}} \tilde{\pi}_j}$$

where the vector $\tilde{\pi}$ is any non-zero solution to $\tilde{\pi} = \tilde{\pi} P$.

4. THE LAGUERRE NUMERICAL LAPLACE TRANSFORM INVERSION TECHNIQUE

In [10], the Laguerre method of [1] (sometimes also referred to as Weeks' method) is modified to derive an efficient algorithm for Laplace transform inversion, which is particularly suited to a parallel implementation. This method represents a function $f(t)$ in terms of its Laplace transform $L(s)$, as the sum

$$f(t) = \sum_{n=0}^{\infty} q_n l_n(t)$$

where:

- the q_n are the *Laguerre coefficients*, given by the Cauchy contour integral

$$q_n = \frac{1}{2\pi i} \int_{C_r} Q(z)/z^{n+1} dz \qquad (2)$$

  In Eq. 2 $Q(z)$ is the *Laguerre generating function* given by

$$Q(z) = \sum_{n=0}^{\infty} q_n z^n = (1-z)L\left(\frac{1+z}{2(1-z)}\right) \qquad (3)$$

  and C_r is a circle about the origin of radius r $(0 < r < 1)$ such that $Q(z)$ is analytic in $\{z : |z| < r\}$.

- the $l_n(t)$ are the *Laguerre functions*, which can be calculated in a numerically stable way from the recursion:

$$l_n(t) = \left(\frac{2n-1-t}{n}\right) l_{n-1}(t) - \left(\frac{n-1}{n}\right) l_{n-2}(t)$$

  starting with $l_0(t) = e^{-t/2}$ and $l_1(t) = (1-t)l_0(t)$, see [1].

Since $|l_n(t)| \leq 1$ for all n, the convergence of the Laguerre series depends on the decay rate of q_n as $n \to \infty$ which is in turn determined by the smoothness of f and its derivatives [1]. Slow convergence of the q_n coefficients can often be improved by exponential dampening and scaling using two real parameters σ and b [17]. Here the inversion algorithm is applied to the function

$$f_{\sigma,b}(t) = e^{-\sigma t} f(t/b)$$

with $f(t)$ being recovered as:

$$f(t) = e^{\sigma b t} f_{\sigma,b}(bt).$$

If no suitable scaling parameters can be found, other inversion methods, such as the Euler method [2] may be used instead. While this is a very robust method that can handle discontinuous functions, it does involve substantially more computation – of the order of 50 distinct evaluations of $L(s)$ for each t point are required.

Traditionally, each q_n coefficient is computed from Eq. 2 by numerical integration, using the trapezoidal rule with $2n$ trapezoids. However, if we apply scaling to ensure that q_n has decayed to (almost) zero by term p_0 (say $p_0 = 200$), we can instead make use of a constant number of $2p_0$ trapezoids when calculating each q_n. This allows us to calculate each q_n with high accuracy while simultaneously providing the opportunity to cache and re-use values of $Q(z)$. Since q_n does not depend on t, and each evaluation of $Q(z)$ involves a single evaluation of $L(s)$, we obtain the response time density at an arbitrary number of t-values at the fixed cost of solving just $2p_0$ linear systems (of the form given in Eq. 1). More details of this approach, including an algorithm for determining suitable scaling parameters, can be found in [10].

5. RESPONSE TIME ANALYSER TOOL

We have implemented a response time analysis pipeline for GSPN models, as shown in Fig. 1. Models are specified in an enhanced form of the DNAmaca Markov Chain Analyser interface language [12, 13]. Appendix A contains an example specification for the simple GSPN model discussed in the results section. From the high-level model, a state space generator produces the reachability graph, along with a list of the source and destination markings that match their respective high-level descriptions. A steady-state solver reads the reachability graph and solves the set of sparse linear equations corresponding to the EMC to compute appropriate weights for the source markings, as described in Section 3.2.

Control is then passed to the distributed Laplace transform inverter, which implements the master-slave model shown in Fig. 1. The inverter is written in C++ and uses the Message Passing Interface (MPI) [9] standard, so it is portable to a wide variety of parallel computers and workstation clusters.

Initially, the master simply runs through our modified Laguerre Laplace transform inversion algorithm of Section 4 and notes the distinct values of s at which $L(s)$ will need to be evaluated. Those values of s for which there is no value of $L(s)$ already stored in the disk cache are added to a global work queue.

At start-up, slave processors read into memory the reachability graph as well as a list of the source and destination markings and the weights to apply to the distribution for each source marking. Each slave processor then applies for an s-value from the global work queue. The slave calculates the corresponding value of $L(s)$ by solving a set of sparse linear equations (of the form given in Eq. 1) using an appropriate iterative numerical method; currently Gauss-Seidel, SOR with dynamic parameter adjustment and Conjugate Gradient Squared (CGS) are supported. Note that, for each different value of s, it is only necessary for a slave processor to modify the diagonal elements of its linear system so set-up is very rapid.

Slave processors return computed values of $L(s)$ to the master. The master stores the returned value in memory and disk caches and immediately issues more work to the slaves if any is available. The disk cache stores values of $L(s)$ using both the value of s and an MD5 checksum of the original high-level model file (as provided by the UNIX utility `md5sum`) as the key. This mechanism avoids redundant work by ensuring that no slave will have to recompute a value of $L(s)$ that has been previously computed for a given model at any time in the past. It also provides a convenient distributed checkpointing mechanism so that parallel jobs that are interrupted can be rapidly restarted without losing work already done.

This master-slave architecture is highly scalable, because the single global work queue with multiple servers ensures a good load balance and very high utilization of slave processors. In addition, there is no inter-slave communication and the amount of master-slave communication required is low and independent of the number of markings.

When all values of $L(s)$ have been computed, the master runs through the Laplace transform inversion algorithm again, this time performing all calculations and obtaining any values of $L(s)$ needed from the memory cache. The resulting points on the response time density curve are written to a disk file, and displayed using the GNUplot graph plotting utility.

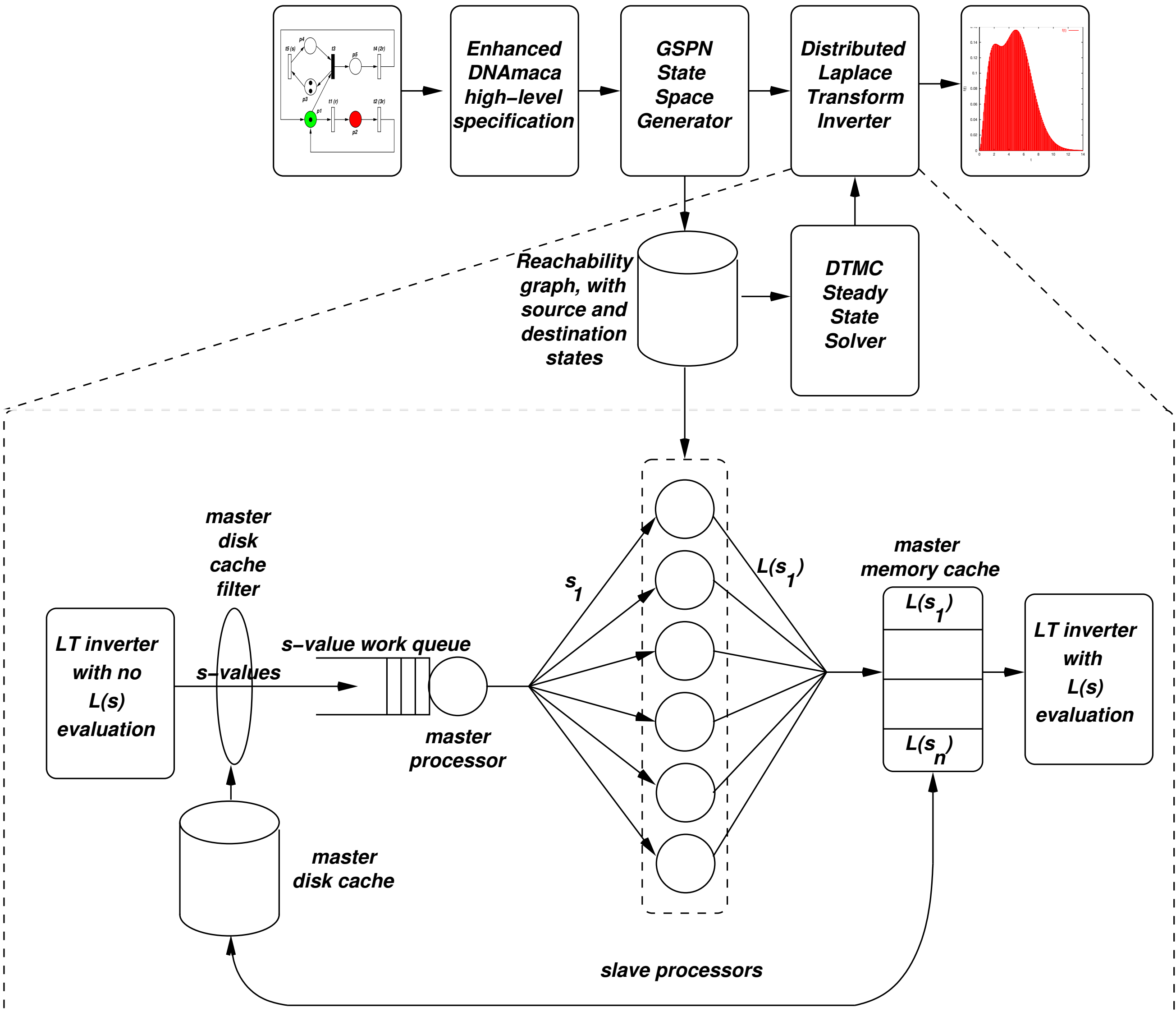

Figure 1: Response time analysis tool showing operation of the distributed Laplace transform inverter in detail

6. CASE STUDIES

6.1 Simple GSPN model

Fig. 2 shows a small contrived GSPN model and its corresponding reachability graph. We illustrate our technique on this GSPN by computing the response time density for the time taken to reach markings where $M(p2) > 0$ from markings where $M(p1) > 0$.

Appendix A shows the enhanced DNAmaca input specification file for this GSPN. The `\model` clause describes the high-level GSPN structure while the `\passage` clause provides a high-level specification of the source and destination markings and the time range over which the performance analyst wishes the resulting response time density to be plotted. Of course, the input file could be constructed automatically by a Petri net tool so that users need only provide the basic information given in the `\passage` clause.

In this example, there are three source markings, two of which are vanishing and one of which is tangible. As discussed in Section 3.2, the Laplace transforms of the passage time from these source markings into the destination markings need to be weighted according to the normalised steady state probabilities of the source markings in the GSPN's EMC. Hence, for source markings $M_1 = (1, 0, 2, 0, 0)$, $M_3 = (1, 0, 1, 1, 0)$ and $M_8 = (1, 0, 0, 2, 0)$,

$$L_{\vec{i}\vec{j}}(s) = \alpha_1 L_{1\vec{j}}(s) + \alpha_3 L_{3\vec{j}}(s) + \alpha_8 L_{8\vec{j}}(s)$$

where $\vec{i} = \{1, 3, 8\}$, $\vec{j} = \{2, 4, 6\}$, $\alpha_1 = 0.22952$, $\alpha_3 = 0.43660$ and $\alpha_8 = 0.33388$ (to 5 s.f.).

Fig. 3 shows the resulting numerical response time density. Also shown is the response time density produced by a discrete-event simulator. There is excellent agreement between the numerical and simulated densities.

For this small example, a single slave processor (a PC with a 1.4GHz Athlon processor and 256MB RAM) required just 18 seconds to calculate the 1600 points plotted on the numerical response time density. No scaling was required and we set $p_0 = 200$, requiring the solution of 402 sets of linear equations (2 of which were necessary to determine that no scaling was required, with the remaining 400 used to compute the q_n coefficients).

6.2 Courier Protocol Software

We now apply our technique to determine an end-to-end response time density in a substantial real-life model. The GSPN shown in Fig. 4 (originally presented in [18]) models the ISO Application, Session and Transport layers of the Courier sliding-window communication protocol. Data flows from a sender ($p1$ to $p26$) to a receiver ($p27$ to $p46$) via a network. The sender's transport layer fragments outgoing data packets; this is modelled as two paths between $p13$ and $p35$. The path via $t8$ carries all fragments before the last one through the network to $p33$. Acknowledgements for these fragments are sent back to the sender (as signalled by the arrival of a token on $p20$), but no data is delivered to the higher layers on the receiver side. The path via $t9$ carries the last fragment of each message block. Acknowledgements for these fragments are generated and a data token is delivered to higher receiver layers via $t27$.

The average number of data packets sent is determined by the ratio of the weights on the immediate transitions $t8$ and $t9$. This ratio, known as the fragmentation ratio, is given by $q1 : q2$ (where $q1$ and $q2$ are the weights associated with transitions $t8$ and $t9$ respectively). This number of data packets is geometrically distributed, with parameter $q1/(q1 + q2)$. In our case study, we use a fragmentation ratio of one.

The transport layer is further characterized by two important parameters: the sliding window size n ($p14$) and the transport space m ($p17$). For our example, we set $m = 1$ and $n = 1$.The transition rates $r1, r2, \ldots, r10$ used in the original model [18] were obtained by benchmarking a working implementation of the protocol. We used rates with the same relative magnitudes, and divided them all by a factor of 5000 to avoid the numerical problems discussed in Section 3.1.

We wished to investigate the end-to-end response time from the initiation of a transport layer transmission to the arrival of the corresponding acknowledgement packet. Consequently we chose as source markings those markings for which $M(p_{11}) > 0$, and as destination markings those for which $M(p_{20}) > 0$. This is appropriate for our sliding window size of $n = 1$ since there can be only one outstanding unacknowledged packet. If we wished to calculate the response time for sliding window sizes greater than one, we would need to augment the state vector used to describe markings to track the progress of a particular token through the Petri net.

The reachability graph contains 29 010 markings, 11 700 of which are tangible and 17 310 of which are vanishing. There are 7 320 source markings and 1 680 destination markings. Fig. 5 shows the resulting numerical response time density. The median (50% quantile) and 95% quantile transmission times are also given. Once again the numerical results are compared against a simulation, and agreement is excellent.

For this example, the Laguerre scaling algorithm [10] selected a dampening parameter of $\sigma = 0.008$ with $p_0 = 200$. A single slave (a 1.4GHz Athlon processor with 256MB RAM) required 24 minutes 15 seconds to calculate the 200 points plotted on the numerical response time density graph. This required the solution of a total of 410 sets of linear equations, 10 of which were needed to determine σ and the remainder used to compute the q_n coefficients. Using 8 slave PCs with the same configuration decreased the required time to just 3 minutes 23 seconds (corresponding to an efficiency of 96%). 16 slave PCs required 2 minutes 17 seconds (72% efficiency). These results reflect the excellent scalability of our approach.

7. CONCLUSION

We have presented an automated numerical technique to compute response time densities in unrestricted Generalised Stochastic Petri net models. A complete response time analysis pipeline has been implemented, including a high-level specification language and a distributed, scalable and checkpointed Laplace transform inverter, based on our own modified Laguerre method. We have applied the pipeline to two case studies, including a realistic model of communication protocol software, and observed excellent agreement with simulation.

The solution of linear systems on slave processors is currently performed in-core, which would limit the solution capacity of individual slave processors to around 3 million states on a 256MB machine. We could easily increase this capacity to around 20 million states by implementing a disk-based solver such as that described in [7]. Further, groups of slave processors could be used to jointly solve very large systems of around 100 million states or more by implementing a parallel disk-based solution method (e.g. [14]).

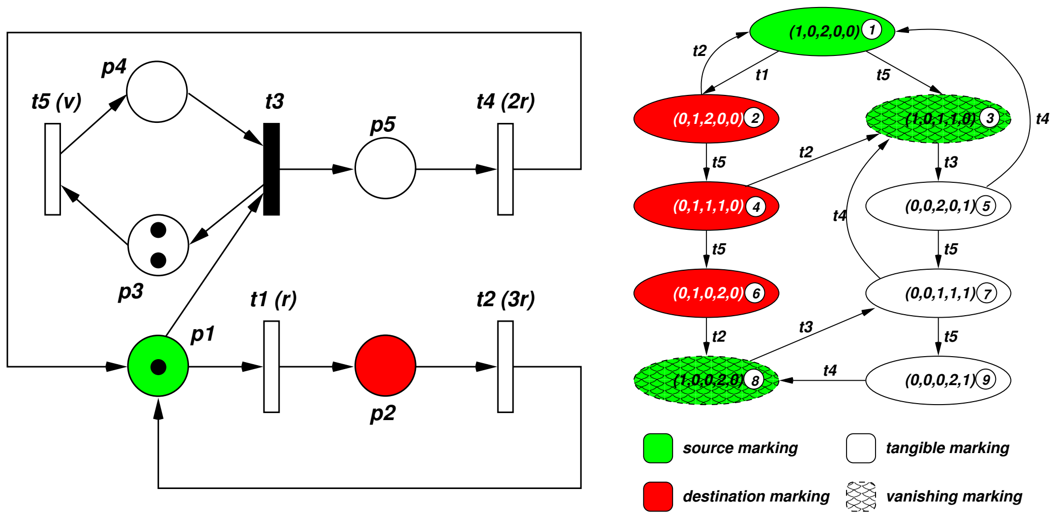

Figure 2: A simple Generalised Stochastic Petri net (left) and its corresponding reachability graph (right).

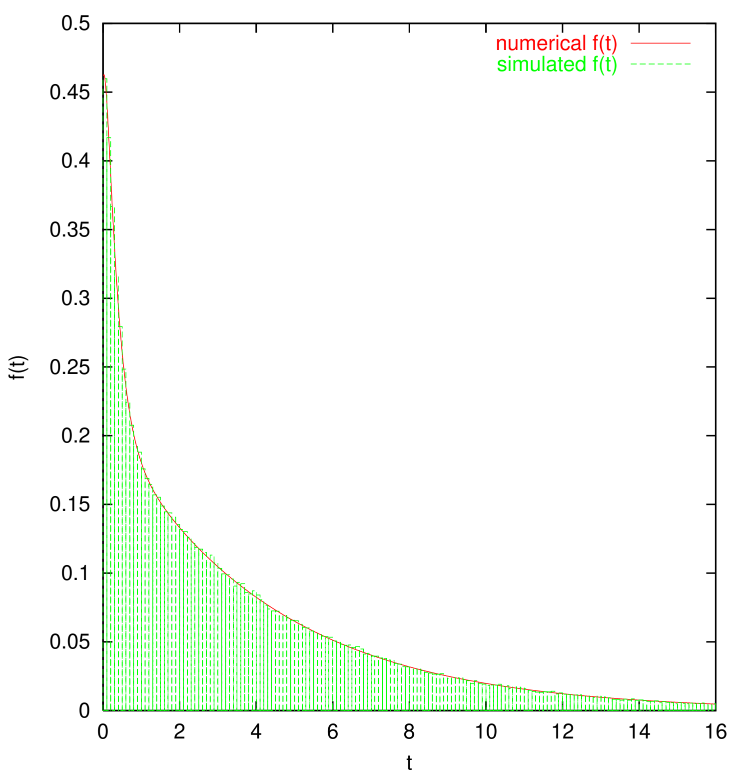

Figure 3: Numerical and simulated response time densities for time taken from markings where $M(p1) > 0$ to markings where $M(p2) > 0$. Here the transition rate parameters are $r = 2$ and $v = 5$.

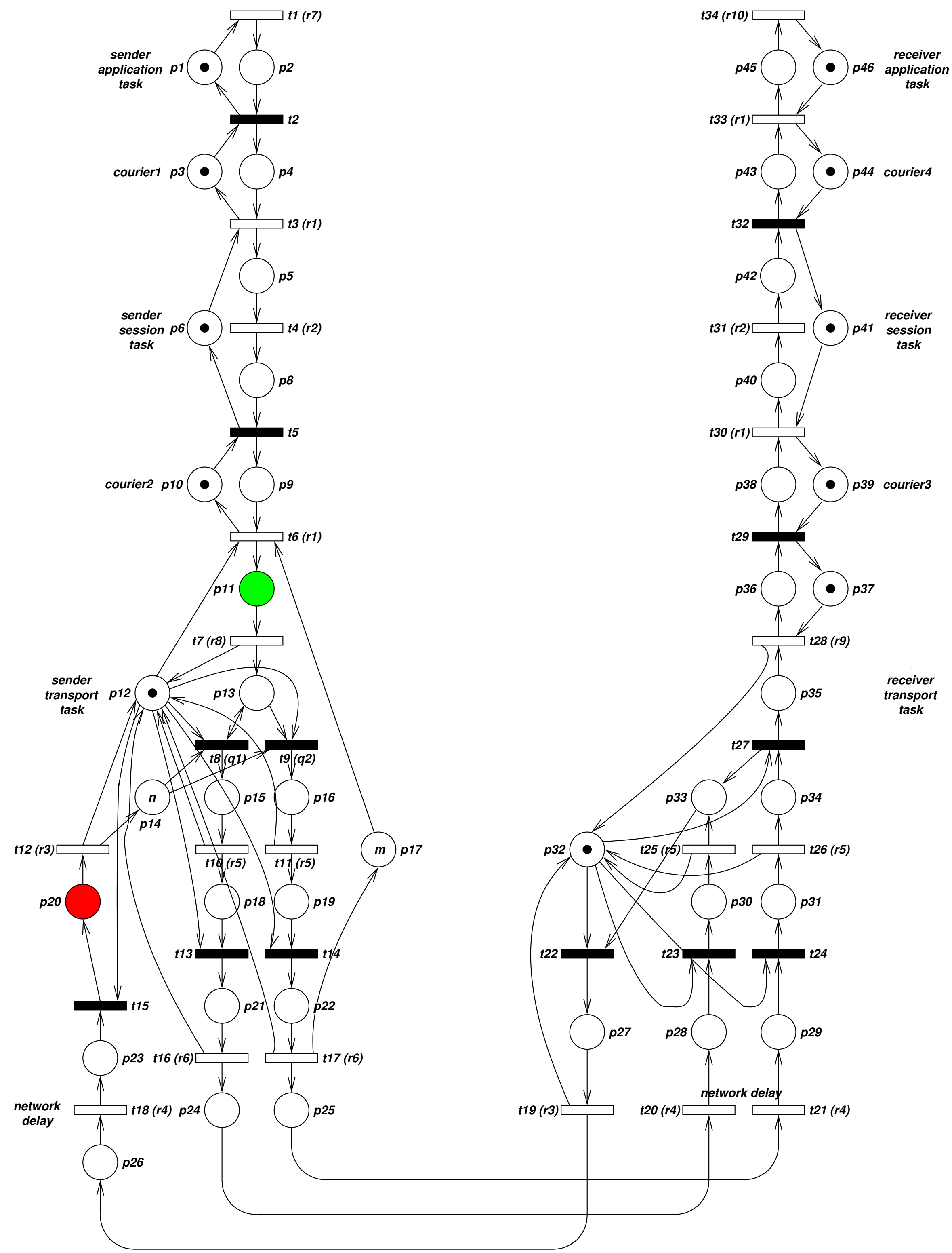

Figure 4: The Courier Protocol Software Generalised Stochastic Petri net.

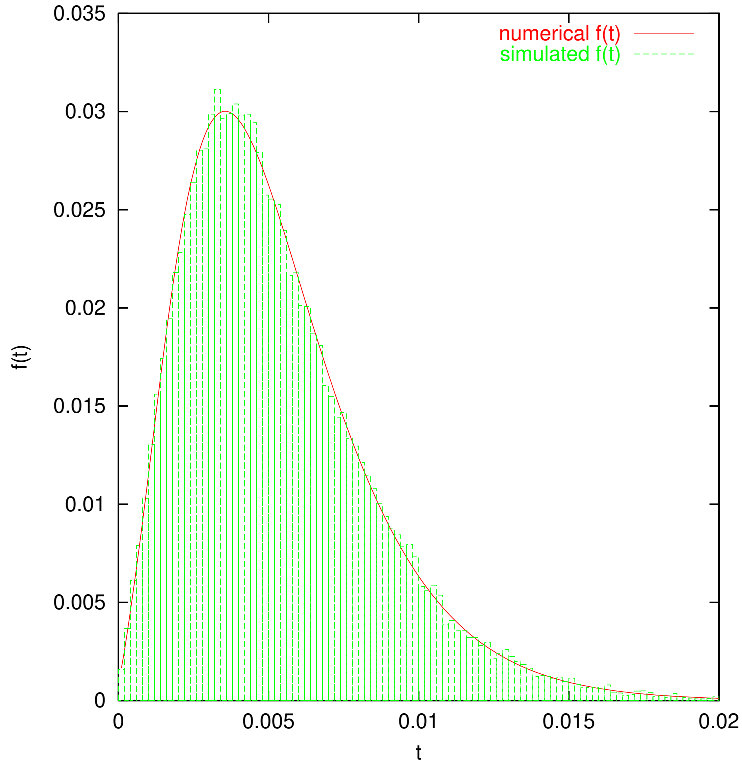

Figure 5: Numerical and simulated response time densities for time taken from the initiation of a transport layer transmission (i.e. those markings for which $M(p_{11}) > 0$) to the arrival of an acknowledgement packet (i.e. those markings for which $M(p_{20}) > 0$). The median response time (50% quantile) is 0.0048 seconds, and the 95% quantile is 0.0114 seconds.

It may be that vanishing states constitute a large proportion of the reachability graph, so that their elimination would be beneficial. This is routine in analyses of equilibrium marking probability distributions [4]. However, in our context, a brute-force application of traditional on-the-fly elimination techniques would destroy our ability to specify vanishing states as source and destination markings (as in the example of Section 6.1). Further, while it has been proved that elimination techniques preserve steady-state probabilities, it needs to be established what further transformations are necessary for response time densities.

Not all response times of interest are simple passage times between markings; they may be related to the progress of a particular token, for example. Such cases require augmentation of the state vector to provide a means for tracking these tokens.

Finally, our performance analysis pipeline could be added as a module to extensible Petri net tools, such as the Petri net Kernel [11] and Medusa [8]. In fact, a module which automatically generates the enhanced model specification shown in Appendix A has already been written for Medusa.

8. ACKNOWLEDGEMENTS

Thanks to Jeremy Bradley for his helpful comments on an earlier draft of this manuscript.

APPENDIX

A. ENHANCED DNAMACA SPECIFICATION

The input specification for the GSPN of Fig. 2 is given below.

```
\model{

  \constant{RR}{2.0}
  \constant{VV}{5.0}

  \statevector{
    \type{short}{ p1, p2, p3, p4, p5 }
  }

  \initial{
    p1 = 1; p3 = 2; p2 = p4 = p5 = 0;
  }

  \transition{t1}{
    \condition{p1 > 0}
    \action{
      next->p1 = p1 - 1; next->p2 = p2 + 1;
    }
    \rate{RR}
  }

  \transition{t2}{
    \condition{p2 > 0}
    \action{
      next->p1 = p1 + 1; next->p2 = p2 - 1;
    }
    \rate{3.0*RR}
  }

  \transition{t3}{
    \condition{p1 > 0 && p4 > 0}
    \action{
      next->p1 = p1 - 1; next->p3 = p3 + 1;
      next->p4 = p4 - 1; next->p5 = p5 + 1;
    }
```

```
    \weight{1.0}
  }

  \transition{t4}{
    \condition{p5 > 0}
    \action{
      next->p1 = p1 + 1; next->p5 = p5 - 1;
    }
    \rate{2.0*RR}
  }

  \transition{t5}{
    \condition{p3 > 0}
    \action{
      next->p3 = p3 - 1; next->p4 = p4 + 1;
    }
    \rate{VV}
  }
}

\passage{
  \source{p1 > 0}
  \destination{p2 > 0}
  \method{laguerre}
  \timerange{0,16,0.01}
}
```

B. REFERENCES

[1] J. Abate, G.L. Choudhury, and W. Whitt. On the Laguerre method for numerically inverting Laplace transforms. *INFORMS Journal on Computing*, 8(4):413–427, 1996.

[2] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems*, 10(1):5–88, 1992.

[3] M. Ajmone-Marsan, G. Conte, and G. Balbo. A class of Generalised Stochastic Petri Nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2:93–122, 1984.

[4] F. Bause and P.S. Kritzinger. *Stochastic Petri net theory*. Verlag Vieweg, Wiesbaden, Germany, 1995.

[5] P. Buchholz, M. Fischer, and P. Kemper. Distributed steady state analysis using Kronecker algebra. In *Proceedings of the 3rd International Meeting on the Numerical Solution of Markov Chains (NSMC '99)*, pages 76–95, Zaragoza, Spain, September 1999.

[6] G. Ciardo and A.S. Miner. A data structure for the efficient Kronecker solution of GSPNs. In *Proceedings of the 8th International Conference on Petri Nets and Performance Models (PNPM '99)*, pages 22–31, Zaragoza, Spain, September 1999. IEEE Computer Society Press.

[7] D.D. Deavours and W.H. Sanders. An efficient disk-based tool for solving very large Markov models. In *Lecture Notes in Computer Science 1245: Proceedings of the 9th International Conference on Modelling, Techniques and Tools (TOOLS '97)*, pages 58–71, St. Malo, France, 3–6 June 1997. Springer Verlag.

[8] N.J. Dingle. Production of the extensible Petri net editor/animator Medusa. Master's thesis, Imperial College, September 2001.

[9] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, Massachussetts, 1994.

[10] P.G. Harrison and W.J. Knottenbelt. Passage time distributions in large Markov chains. In *Proc. ACM SIGMETRICS 2002*, Marina Del Rey, California, June 2002.

[11] E. Kindler and M. Weber. The Petri Net Kernel. In K. H. Mortensen, editor, *Tool Demonstrations at the 21st International Conference on the Theory and Application of Petri nets*, pages 71–75. Aarhus, Denmark, June 2000.

[12] W.J. Knottenbelt. Generalised Markovian analysis of timed transition systems. Master's thesis, University of Cape Town, Cape Town, South Africa, July 1996.

[13] W.J. Knottenbelt. *Parallel Performance Analysis of Large Markov Models*. PhD thesis, Imperial College, London, United Kingdom, February 2000.

[14] W.J. Knottenbelt and P.G. Harrison. Distributed disk-based solution techniques for large Markov models. In *Proceedings of the 3rd International Meeting on the Numerical Solution of Markov Chains (NSMC '99)*, pages 58–75, Zaragoza, Spain, September 1999.

[15] J.L. Peterson. *Petri Nets and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[16] W. Reisig. *A Primer in Petri Net Design*. Springer Verlag, 1992.

[17] W.T. Weeks. Numerical inversion of Laplace transforms using Laguerre functions. *Journal of the ACM*, 13:419–426, 1966.

[18] 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*, pages 64–73, Melbourne, Australia, 2–5 December 1991. IEEE Computer Society Press.