Blending Randomness in Closed Queueing Network Models

Giuliano Casale

Department of Computing, Imperial College London, SW7 2AZ, London, UK, g.casale@imperial.ac.uk

Mirco Tribastone

Electronics and Computer Science, University of Southampton, United Kingdom, m.tribastone@soton.ac.uk

Peter G. Harrison

Department of Computing, Imperial College London, SW7 2AZ, London, UK, pgh@doc.ic.ac.uk

Abstract

Random environments are stochastic models used to describe events occurring in the environment a system operates in. The goal is to describe events that affect performance and reliability such as breakdowns, repairs, or temporary degradations of resource capacities due to exogenous factors. Despite having been studied for decades, models that include both random environments and queueing networks remain difficult to analyse. To cope with this problem, we introduce the *blending algorithm*, a novel approximation for closed queueing network models in random environments. The algorithm seeks to obtain the stationary solution of the model by iteratively evaluating the dynamics of the system in between state changes of the environment. To make the approach scalable, the computation relies on a fluid approximation of the queueing network model. A validation study on 1,800 models shows that blending can save a significant amount of time compared to simulation, with an average accuracy that grows with the number of servers in each station. We also give an interpretation of this technique in terms of Laplace transforms and use this approach to determine convergence properties.

Key words: Random environments, fluid models, iterative approximation, transient analysis, Laplace transform

1 Introduction

The recent growth of interest in cloud computing has lead researchers to investigate systems that operate in environments shared by multiple tenants. Modelling these environments is challenging, since performance and availability levels for a user may change over time in a complex manner depending on the activity of the other tenants. For example, cloud virtual machines may experience transient CPU contention periods due to the activity of other virtual machines co-located on the same physical host. Similarly, a web farm deployed on the cloud may experience network bandwidth fluctuations due to traffic generated by other tenants. Accurate models of these systems require the ability to characterise exogenous events that arise in the operational environment of a system.

In performance and availability prediction, the problem of environment modelling may be tackled by associating a state, called a *stage*, to each possible condition of the environment. The active stage of the environment is assumed to evolve in time according to a Markov process. The description of the system then becomes parametric in the currently active stage. A model of this kind is said to describe the system as operating in a *random environment*. Queueing systems operating in random environments have been investigated for decades [30, 33], however models involving queueing networks are often intractable, calling for the development of customised performance evaluation methodologies.

Here we focus on systems that can be modelled as closed queueing network models, for example multi-tier software systems [42]. There is a limited literature on closed systems operating in random environments, often because they do not enjoy closed form expressions of the state probabilities except in particular cases [17, 20, 44]. In comparison, several works have focused on open systems operating in random environments, which are often tractable by means of conditional PASTA arguments [23]. The lack of results for the closed case limit their tractability to small problems where the underlying Markov model can be evaluated numerically. Alternatively, simulation may be used, but its running time makes it impractical for optimisation studies.

A computationally more viable route is offered by analytical approximations of closed networks based on product-form queueing theory. In the *average-environment* approximation [3], the system is studied with exponential state transitions having rates equal to their average value when the environment reaches equilibrium. This captures well environments where transitions happen *frequently* with respect to the representative timescales of the system. Conversely, in the *decomposition* approximation, an isolated model is evaluated for each stage of the random environment. The isolated solutions are

then weighted across stages using the equilibrium distribution of the random environment [14,43]. This approach describes well an environment where transitions happen *rarely* compared to the representative timescales of the system (e.g., component failures). However, in intermediate situations both types of approximations tend to be loose. While numerical iterative methods exist to address this type of issue [16], they are unable to cope with state spaces of the scale considered in large queueing network models. This calls for developing approximation techniques for queueing network analysis in random environments that are more robust than average-environment and decomposition approximations.

To cope with this challenge, we present the *blending* algorithm, an iterative approximation scheme for closed queueing networks operating in random environments. Blending evaluates at each iteration the dynamics of the system in between two successive stage changes using transient analysis. A sequence of these analyses is used to determine a fixed point for the transient trajectories. The fixed point is then used to compute performance measures of the system for all stages of the random environment. From such embedded averages, it is simple to approximate the long-term equilibrium behaviour of the system and thus estimate its performance measures. We also interpret this value as a Laplace transform evaluated at a specific point and use this interpretation to develop a convergence analysis for the method.

The main challenge of the blending approach is to represent the transient dynamics of the system in between two successive stage changes. Since queueing network models often have an intractably large state space, we cannot use direct numerical methods to evaluate the underlying Markov process. To cope with state space explosion, we use a fluid approximation based on ordinary differential equations (ODEs) defined in the sense of Kurtz [28]. Even though Kurtz's theory has found a wealth of applications in a broad range of disciplines, including chemistry [29], ecology [35], randomised algorithms [32], and communication networks [10], the application to closed models in random environments seems novel. Using Kurtz's theory, we analyse the queueing network as a density-dependent process. This process is used to approximate inexpensively the transient evolution of queue-lengths in between successive stage changes. The number of ODEs does not increase with the number of jobs, which is the parameter that most influences the state space size, thus enabling the evaluation of large-scale models.

We evaluate the algorithm by a comprehensive numerical study consisting of 1,800 models. We solve these models by the blending algorithm and compare results with simulation. The study is quite extensive, corresponding to a month of computation time on a private cluster. The results for blending are in good agreement with simulation, but are far less expensive computationally.

The rest of this paper is organised as follows. Section 2 provides a motivating example. The class of models we focus on is defined in Section 3. The high-level structure of the proposed algorithm is defined in Section 4 and further developed in Section 5, where we introduce a fluid approximation for the class of models considered. In Section 6 we develop the blending algorithm and discuss its convergence in Section 7. Numerical results on random instances are given in Section 8. Related work is discussed in Section 9, followed by conclusions in Section 10. Appendix A provides related material on Markovian random environments.

2 Motivation

To motivate the present work, we present a toy example illustrating the effects of a random environment on the performance measures of a closed queueing network model. The interested reader can find similar examples in previous work, e.g., for open models [23]. The goal of this motivating example is to build intuition on the type of transient behaviours that a closed network can exhibit in the presence of a random environment. As we show in Appendix A, the statements in this section hold for a more general class of systems than queueing networks.

Consider a tandem closed network composed of a first-come first-served (FCFS) single-server queue and a delay station (infinite-server queue, $-/GI/\infty$). The network topology is cyclic, with a population of $N = 2$ jobs. Service times at the single-server queue are exponentially distributed with rate μ. The queueing network operates in a Markovian random environment that causes server breakdown at the single-server queue with rate α_{21}, and subsequent repair with rate α_{12}. The random environment is independent of the state of the stations. Before breakdown, the single-server queue processes jobs with rate $\sigma_2 > 0$, but upon a breakdown it switches to a rate $\sigma_1 < \sigma_2$ until repaired. This simple system may be described by a continuous-time Markov chain (CTMC) with infinitesimal generator $\boldsymbol{Q}$ embedding an environment with generator $\boldsymbol{E}$, where

$$\boldsymbol{Q} = \left[\begin{array}{c|c} \boldsymbol{Q}_1^* - \alpha_{12}\boldsymbol{I} & \alpha_{12}\boldsymbol{I} \\ \hline \alpha_{21}\boldsymbol{I} & \boldsymbol{Q}_2^* - \alpha_{21}\boldsymbol{I} \end{array}\right], \qquad \boldsymbol{E} = \begin{bmatrix} -\alpha_{12} & \alpha_{12} \\ \alpha_{21} & -\alpha_{21} \end{bmatrix},$$

$$\boldsymbol{Q}_1^* = \begin{bmatrix} -2\mu & 2\mu & 0 \\ \sigma_1 & -(\sigma_1+\mu) & \mu \\ 0 & \sigma_1 & -\sigma_1 \end{bmatrix}, \qquad \boldsymbol{Q}_2^* = \begin{bmatrix} -2\mu & 2\mu & 0 \\ \sigma_2 & -(\sigma_2+\mu) & \mu \\ 0 & \sigma_2 & -\sigma_2 \end{bmatrix}.$$

Each state (row) of $\boldsymbol{Q}$ represents a reachable state of the system. The matrices

$\boldsymbol{Q}_1^*$ and $\boldsymbol{Q}_2^*$ are the infinitesimal generators for the system conditional on the environment being in stage 1 or 2, respectively. All Markov processes considered in this paper are irreducible and ergodic.

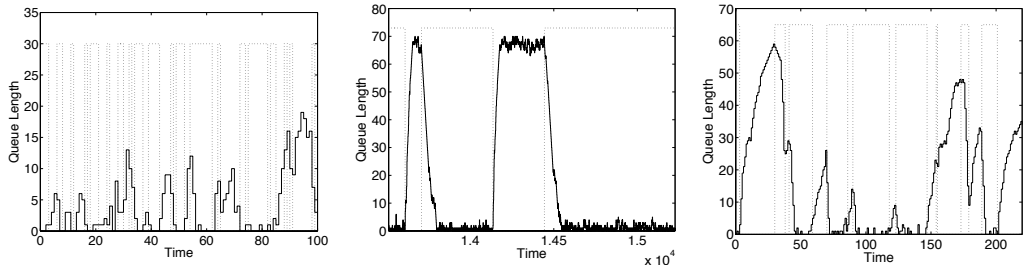
To illustrate the challenges in analysing random environments, we recall that the *average-environment approximation* for the model under study can be obtained by solving the Markov process with generator $\boldsymbol{Q}^{AVG} = \pi_1^{env}\boldsymbol{Q}_1^* + \pi_2^{env}\boldsymbol{Q}_2^*$, where $\pi_1^{env} = \frac{\alpha_{21}}{\alpha_{12}+\alpha_{21}}$ is the equilibrium probability of stage 1 in the random environment. Conversely, the *decomposition approximation* weights by π_1^{env} and $\pi_2^{env} = 1 - \pi_1^{env}$ the equilibrium solutions of the generators $\boldsymbol{Q}_1^*$ and $\boldsymbol{Q}_2^*$ considered in isolation, respectively.

We first consider this model with service rate $\mu = 0.1$ at the delay station, $\sigma_1 = 1.0$, and $\sigma_2 = 10.0$ for the single-server queue. The job population is set to $N = 40$. Figure 1 illustrates queue-length sample paths for different values of $\alpha = \alpha_{12} + \alpha_{21}$, which is called the environment event rate.

Figure 1(a) shows the case of a high event rate with $\alpha_{12} = 2$ and $\alpha_{21} = 1$. In this case, the average-environment approximation is very accurate, see Appendix A for a mathematical justification.

Figure 1(b) shows the opposite case where changes in the environment occur rarely. Here we set $\alpha_{12} = 0.0002$ and $\alpha_{21} = 0.0001$. Note that the time scale represented on the horizontal axis is much longer than that in the other plots since stage transitions happen infrequently. Upon arrival of an event that moves the environment into stage 1, the network enters a long transient period where jobs migrate from the single-server queue to the delay station. Instead, upon a jump to stage 2, the jobs quickly accumulate in the single-server queue. For models of this kind, decomposition is known to be accurate because the length of the transient period is quite short compared to the overall holding time in a stage [7].

Figure 1(c) considers $\alpha_{12} = 0.02$ and $\alpha_{21} = 0.01$, i.e., comparable time scales between environmental events and service rates. Similarly to the decomposition model in Figure 1(c), the network cyclically enters transient periods where jobs migrate from one station to the other. However, the length of such transients is often comparable to the sojourn time of the random environment in each stage. Thus, the queue-length value observed at the instant of a stage change is a random variable that is tightly coupled with the sojourn time in each stage. In this situation, it is easy to show examples where neither average-environment nor decomposition are accurate. We show in Appendix A that linear combinations of the results of the two approximations do not provide a solution to this problem. This motivates our interest for developing novel analysis methods for the problem under study.



(a) High environment transition rate.

(b) Low environment transition rate.

(c) Intermediate environment transition rate.

Fig. 1. Queue-length sample paths in a random environment

3 Model

We now introduce relevant notation and assumptions. The random environments considered in this paper evolve according to a CTMC jumping between stages. The environment is specified by the following parameters:

- E: number of stages;
- $e, h = 1, \ldots, E$: stage indices;
- α_{eh}: rate of jump from stage e to stage h;
- $\alpha_e = \sum_{h \neq e} \alpha_{eh}$: total outgoing rate from stage e;
- π_h^{env}: probability of the environment being in stage h at equilibrium

Using the above definitions, the rates of the environment can be organised in an infinitesimal generator

$$\boldsymbol{E} = \begin{bmatrix} -\alpha_1 & \alpha_{12} & \ldots & \alpha_{1E} \\ \alpha_{21} & -\alpha_2 & \ldots & \alpha_{2E} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{E1} & \alpha_{E2} & \ldots & -\alpha_E \end{bmatrix} \tag{1}$$

In this paper, we consider only queueing networks processing a constant number of jobs, i.e., closed queueing networks. We let N be the number of circulating jobs in the network, and denote by M the number of stations. Both N and M are assumed to be independent of the currently active stage of the environment. All stations have unbounded buffer capacity and FCFS service discipline. Service times are independent and identically distributed random variables following a Coxian distribution, which includes exponential, hyperexponential, and Erlang distributions as special cases [7]. We shall refer to the

states of the Coxian service processes as *phases*.

When the random environment is in stage e, the queueing network is parameterised as follows:

- $i, j = 1, \ldots, M$: station indices;
- $r^e_{i,j}$: routing probability from station i to station j;
- S^e_i: number of servers at station i;
- K^e_i: number of phases in the Coxian service process at station i;
- $k \equiv k(i) = 1, \ldots, K^e_i$: phase index for the Coxian service process at station i;
- $\mu^e_{i,k}$: service rate in phase $k \equiv k(i)$ at station i;
- $\phi^e_{i,k}$: probability for jobs at station i to complete after receiving service in phase $k \equiv k(i)$.

We shall omit the dependence of k upon the station index i whenever clear from the context.

Finally, we assume that upon a change of stage in the environment, the state of each station is instantaneously modified by moving all active jobs to the first phase of service. Note that exponential servers are insensitive to this rule, since they have a single phase of service. We refer to this assumption as the *reset rule*. Although the reset rule is a technical device used for mathematical convenience, it does describe correctly some systems – for example those where a repair after a breakdown leads to restarting the execution of all jobs in a system. Extensions of the blending methodology to other rules appear possible, but we leave this research for future work [1] .

Running Example. *Throughout the paper, we use the following running example to illustrate our methodology: a modification of the tandem network of Section 2. As before, the delay server has rate* $\mu^1_{1,1} = \mu^2_{1,1} = \mu$ *in all stages. However, the FCFS queue now has* $S^1_2 = S^2_2 = S_2$ *servers in all stages: in stage 1 its service times are exponential with rate* $\mu^1_{2,1} = \sigma_1$*; in stage 2 they are Erlang-2 with rates* $\mu^2_{2,1} = \mu^2_{2,2} = \sigma_2$*. The model is parametric in* S_2 *and the job population* N*. Throughout the paper, we assume* $\mu = 1$*;* $\sigma_1 = 2$*,* $\sigma_2 = 3$*,* $\alpha_{12} = 4$*,* $\alpha_{21} = 5$*.*

[1] For example, if the service process at a station is independent of the active stage of the environment, one may simply want to avoid altering job phases when the environment changes. This feature is already supported for servers with exponential service time, but the extension to Coxian service processes requires a minor modification of the blending algorithm by changing the definition of the $\boldsymbol{R}_{he}$ matrices. Early experiments suggest that blending would continue to work under this rule.

From the above definitions, the queueing network model may be readily described by a CTMC having state vector $\boldsymbol{Y} = (\boldsymbol{n}, e)$, where $e = 1, \ldots, E$ denotes the current stage of the random environment. The distribution of jobs across stations is described by the vector $\boldsymbol{n} = (\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_M)$, $\boldsymbol{n}_i$ being the state vector of station i, further specified as $\boldsymbol{n}_i = (x_i, s_{i,1}, \ldots, s_{i,K_i^e})$, where:

- x_i is the sum of two terms, the number of jobs waiting in the queue buffer and the number of jobs that are currently receiving service in (Coxian) phase 1. This representation simplifies the fluid analysis equations in Section 5.
- $s_{i,k}$ is the number of servers that are currently processing jobs in phase k, thus $\sum_{k=1}^{K_i^e} s_{i,k} = S_i^e$. If a server is idle, it is counted within $s_{i,1}$ together with the servers processing jobs in phase 1. If i is a delay station, we set $S_i^e = +\infty$. [2]

The above definitions imply that the total number of jobs at station i in any state is

$$n_i = x_i + \sum_{k=2}^{K_i^e} s_{i,k} \tag{2}$$

for all stages $e = 1, \ldots, E$.

The CTMC defined on the proposed state space may be specified using a collection of *transition rate functions*. These are functions that provide the transition rate between any pair of states $(\boldsymbol{n}, e)$ and $(\boldsymbol{n}', h)$ that are connected by a transition in the CTMC. For the class of models we consider, only five transition rate functions are needed to specify the CTMC, they are shown in Table 1. The left column provides the service rate definition and its dependence on the current state $(\boldsymbol{n}, e)$ and other parameters specific to the transition. The right column describes the destination state $\boldsymbol{Y}' = (\boldsymbol{n}', h)$ after the transition occurs.

A description of the transition rate functions shown in Table 1 is as follows:

- $f_1^{dep}(\boldsymbol{n}, e, i, j)$ describes the rate at which jobs in station i complete after the first phase of service and are then routed to station j, conditional on the environment being in stage e. The case $i = j$ is excluded since the associated transition does not change the state of any station in the network.
- $f_k^{dep}(\boldsymbol{n}, e, i, j)$ is similar to $f_1^{dep}(\boldsymbol{n}, e, i, j)$, but provides the rate for completions at the kth phase of service. The case $i = j$ is here valid since completions returning at the same station restart from a different phase, phase 1, thus changing the state of the station.

[2] In implementations, this value may be set equal to the total job population N.

Transition rate	*Destination state* $\boldsymbol{Y}'$
$f_1^{dep}(\boldsymbol{n}, e, i, j) = \phi_{i1}^e r_{ij}^e \mu_{i1}^e \min(x_i, s_{i1})$ for $i, j = 1, \ldots, M; i \neq j$, $e = 1, \ldots, E$	$\boldsymbol{Y}' = (\boldsymbol{n}', e) = (\boldsymbol{n}_1, \ldots, \boldsymbol{n}_i', \ldots, \boldsymbol{n}_j', \ldots, \boldsymbol{n}_M, e)$ $\boldsymbol{n}_i' = (x_i - 1, s_{i1}, s_{i2}, \ldots, s_{iK_i^e})$ $\boldsymbol{n}_j' = (x_j + 1, s_{j1}, s_{j2}, \ldots, s_{jK_i^e})$
$f_k^{dep}(\boldsymbol{n}, e, i, j) = \phi_{ik}^e r_{ij}^e \mu_{ik}^e s_{ik}$ for $i, j = 1, \ldots, M$, $e = 1, \ldots, E$ $k = 2, \ldots, K_i^e$	$\boldsymbol{Y}' = (\boldsymbol{n}', e) = (\boldsymbol{n}_1, \ldots \boldsymbol{n}_i', \ldots, \boldsymbol{n}_j', \ldots, \boldsymbol{n}_M, e)$ $\boldsymbol{n}_i' = (x_i, s_{i1} + 1, \ldots, s_{ik} - 1, \ldots, s_{iK_i^e})$ $\boldsymbol{n}_j' = (x_j + 1, s_{j1}, s_{j2}, \ldots, s_{jK_i^e})$
$f_1^{ph}(\boldsymbol{n}, e, i) = (1 - \phi_{i1}^e)\mu_{i1}^e \min(x_i, s_{i1})$ for $i = 1, \ldots, M$, $e = 1, \ldots, E$	$\boldsymbol{Y}' = (\boldsymbol{n}', e) = (\boldsymbol{n}_1, \ldots \boldsymbol{n}_i', \ldots, \boldsymbol{n}_M, e)$ $\boldsymbol{n}_i' = (x_i - 1, s_{i1} - 1, s_{i2} + 1, \ldots, s_{iK_i^e})$
$f_k^{ph}(\boldsymbol{n}, e, i) = (1 - \phi_{ik}^e)\mu_{ik}^e s_{ik}$ for $i = 1, \ldots, M$, $e = 1, \ldots, E$, $k = 2, \ldots, K_i^e$	$\boldsymbol{Y}' = (\boldsymbol{n}', e) = (\boldsymbol{n}_1, \ldots \boldsymbol{n}_i', \ldots, \boldsymbol{n}_M, e)$ $\boldsymbol{n}_i' = (x_i, s_{i1}, \ldots, s_{ik} - 1, s_{i(k+1)} + 1, \ldots, s_{iK_i^e})$
$f^{env}(\boldsymbol{n}, e, h) = \alpha_{eh}$ for $e, h = 1, \ldots, E, e \neq h$	$\boldsymbol{Y}' = (\boldsymbol{n}', h) = (\boldsymbol{n}_1', \ldots, \boldsymbol{n}_M', h)$ $\boldsymbol{n}_i' = \left(x_i + \sum_{k=2}^{K_i^e} s_{ik}, s_{i1} + \sum_{k=2}^{K_i^e} s_{ik}, \underbrace{0, \ldots, 0}_{K_i^e - 1}\right)$

Table 1
Transition rate functions. Source state is $(\boldsymbol{n}, e) = (\boldsymbol{n}_1, \ldots, \boldsymbol{n}_i, \ldots, \boldsymbol{n}_M, e)$.

- $f_1^{ph}(\boldsymbol{n}, e, i)$ describes the rate at which servers in station i advance from the first to the second Coxian phase, conditional on the environment being in stage e.
- $f_k^{ph}(\boldsymbol{n}, e, i)$ is similar to $f_1^{ph}(\boldsymbol{n}, e, i)$, but refers to the rate at which jobs in service advance from the kth phase to $(k+1)$th phase.
- $f^{env}(\boldsymbol{n}, e, h)$ describes the rate at which the random environment stage changes from stage e to stage $h \neq e$.

Using the transition rate functions in Table 1, the infinitesimal generator of the model as a whole may be written as

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_{11} & \cdots & \boldsymbol{Q}_{1e} & \cdots & \boldsymbol{Q}_{1E} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{Q}_{e1} & \cdots & \boldsymbol{Q}_{ee} & \cdots & \boldsymbol{Q}_{eE} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{Q}_{E1} & \cdots & \boldsymbol{Q}_{Ee} & \cdots & \boldsymbol{Q}_{EE} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{Q}_{ee} = \boldsymbol{Q}_e^* - \alpha_e \boldsymbol{I}$, $\boldsymbol{Q}_e^*$ is the infinitesimal generator of the queueing network model considered in isolation with the parameterisation for stage e, $\boldsymbol{I}$ is the identity matrix, and for $e \neq h$, $\boldsymbol{Q}_{eh} = \alpha_{eh} \boldsymbol{R}_{eh}$, where $\boldsymbol{R}_{eh}$ is a

matrix encoding the reset rule. Matrix $\boldsymbol{R}_{eh}$ has a 1 at each row-index pair (i, j) corresponding to pair of states $\boldsymbol{Y} = (\boldsymbol{n}, e)$ and $\boldsymbol{Y}' = (\boldsymbol{n}', h)$ mapped by $f^{env}(\boldsymbol{n}, e, h)$ in Table 1; all the other entries are 0.

Running Example. *According to the previous definitions, the state descriptor for the running example is*

$$\boldsymbol{Y} = \begin{cases} \boldsymbol{n} = ((x_1, s_{1,1}), (x_2, s_{2,1})), & x_1 + x_2 = N, s_{1,1} = +\infty, s_{2,1} = S_2, & \text{if } e = 1 \\ \boldsymbol{n} = ((x_1, s_{1,1}), (x_2, s_{2,1}, s_{2,2})) & x_1 + x_2 = N, s_{1,1} = +\infty, s_{2,1} + s_{2,2} = S_2, & \text{if } e = 2 \end{cases}$$

For readability, we omit the constant terms in the above expressions, so that we may write $\boldsymbol{n} = (x_1, x_2)$ *in stage* $e = 1$ *and* $\boldsymbol{n} = (x_1, x_2, s_{2,1}, s_{2,2})$ *in stage* $e = 2$*. Assume for example* $N = 2$ *and* $S_2 = 1$ *such that in stage* 1 *it is*

$$\boldsymbol{n} \in \{(2, 0), (1, 1), (0, 2)\}$$

and in stage 2 it is

$$\boldsymbol{n} \in \{(2, 0, 1, 0), (1, 1, 1, 0), (1, 1, 0, 1), (0, 2, 1, 0), (0, 2, 0, 1)\}.$$

Given this state ordering, the CTMC of the model has generator

$$\boldsymbol{Q} = \left[\begin{array}{c|c} \boldsymbol{Q}_1^* - \alpha_{12}\boldsymbol{I} & \alpha_{12}\boldsymbol{R}_{12} \\ \hline \alpha_{21}\boldsymbol{R}_{21} & \boldsymbol{Q}_2^* - \alpha_{21}\boldsymbol{I} \end{array}\right],$$

where

$$\boldsymbol{Q}_1^* = \begin{bmatrix} -2\mu & 2\mu & 0 \\ \sigma_1 & -(\sigma_1 + \mu) & \mu \\ 0 & \sigma_1 & -\sigma_1 \end{bmatrix}, \qquad \boldsymbol{R}_{12} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\boldsymbol{R}_{21} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{Q}_2^* = \begin{bmatrix} -2\mu & 2\mu & 0 & 0 & 0 \\ 0 & -(\sigma_2 + \mu) & \sigma_2 & \mu & 0 \\ \sigma_2 & 0 & -(\sigma_2 + \mu) & 0 & \mu \\ 0 & 0 & 0 & -\sigma_2 & \sigma_2 \\ 0 & \sigma_2 & 0 & 0 & -\sigma_2 \end{bmatrix}.$$

Notice in particular that $\boldsymbol{R}_{12}$ *and* $\boldsymbol{R}_{21}$ *reset all FCFS queue servers to phase 1, according to the reset rule.*

We now illustrate the transition rates for the system in stage 2 using the entries of $\boldsymbol{Q}_2^*$. Let $\boldsymbol{Q}_2^*(i,j)$ be the entry of $\boldsymbol{Q}_2^*$ on row i and column j. The transition rates are then:

- $f_1^{dep}(\boldsymbol{n},2,1,2) = \mu_{1,1}^2 \min(x_1,+\infty) = \mu x_1$*, e.g.,* $\boldsymbol{Q}_2^*(1,2)$, $\boldsymbol{Q}_2^*(2,4)$, $\boldsymbol{Q}_2^*(3,5)$.
- $f_1^{dep}(\boldsymbol{n},2,2,1) = 0$ *since an Erlang-2 server cannot complete in phase 1 (*$\phi_{2,1}^2 = 0$*), e.g.,* $\boldsymbol{Q}_2^*(2,1)$, $\boldsymbol{Q}_2^*(4,3)$.
- $f_2^{dep}(\boldsymbol{n},2,2,1) = \mu_{2,2}s_{2,2} = \sigma_2 s_{2,2}$*,since an Erlang-2 server always completes in phase 2, e.g.,* $\boldsymbol{Q}_2^*(3,1)$, $\boldsymbol{Q}_2^*(5,2)$.
- $f_1^{ph}(\boldsymbol{n},2,2) = \mu_{2,1}\min(x_2,s_{2,1}) = \sigma_2\min(x_2,s_{2,1})$ *is the rate of servers moving from the first to the second Erlang stage, e.g.,* $\boldsymbol{Q}_2^*(2,3)$, $\boldsymbol{Q}_2^*(4,5)$.

4 Iterative Analysis at Stage Transition Times

The CTMC described in the previous section suffers state explosion issues as the number of queues, jobs, and service phases in the model increases. To cope with the state space explosion issue, we characterise the equilibrium behaviour of the model by focusing on the network state observed immediately after a stage transition occurs in the random environment. Specifically, we focus on the average queue-lengths seen at instants of stage change in the random environment.

We begin with some definitions. We consider the system at equilibrium and assume to observe it at an instant immediately *after* the environment enters stage h. We call *entry probability vector* for stage h the vector of equilibrium state probabilities for the system seen at this instant and denote it by $\boldsymbol{\eta}^h$. After spending an exponentially distributed time with rate α_h in stage h, the environment moves to a new stage $e \neq h$. We call *exit probability vector* for stage h, the vector of equilibrium state probabilities seen immediately *before* this instant and denote it by $\boldsymbol{\xi}^h$. Then, by construction

$$\boldsymbol{\xi}^h = \int_0^{+\infty} \boldsymbol{\eta}^h e^{\boldsymbol{Q}_h^* t} \alpha_h e^{-\alpha_h t} dt, \tag{4}$$

where $\alpha_h e^{-\alpha_h t}$ is the probability that the environment leaves stage h after t time units and the vector $\boldsymbol{\eta}^h e^{\boldsymbol{Q}_h^* t}$ gives the system state probabilities after t time units since entering stage h, as determined by the Kolmogorov forward equations of the underlying Markov process, conditional on the system staying in stage h along the whole sample path.

We now further investigate the relationship between entry and exit probability vectors. Consider the exit probability vector $\boldsymbol{\xi}^h$ for a stage h and assume that the environment next evolves to a stage $e \neq h$ at some instant. Then,

to determine the entry probability vector for stage e conditional on the destination stage e, $\boldsymbol{\xi}^h$ needs to be multiplied by $\boldsymbol{R}_{he}$ that applies the reset rule for this stage transition. Removing the conditioning on the destination stage, the entry probability vector $\boldsymbol{\eta}^e$ will be determined as an average over all the vectors $\boldsymbol{\xi}^h \boldsymbol{R}_{he}$ for all jumps originating from stages $h \neq e$. That is, we can write

$$\boldsymbol{\eta}^e = \sum_{\substack{h=1 \\ h \neq e}}^{E} p_{he}^{src} \boldsymbol{\xi}^h \boldsymbol{R}_{he},$$

where p_{he}^{src} is the probability of stage h being the source of the jump to e. This probability term can be readily obtained by standard arguments as

$$p_{he}^{src} = \frac{\pi_h^{env} \alpha_{he}}{\sum_{j \neq e} \pi_j^{env} \alpha_{je}}, \qquad \sum_{h \neq e} p_{he}^{src} = 1, \tag{5}$$

Equations (4)-(5) immediately imply the following proposition.

Proposition 1 *In the CTMC with generator* (3)*, at equilibrium, the probability vectors embedded at stage entry instants satisfy the balance equation*

$$\boldsymbol{\eta}^e = \sum_{\substack{h=1 \\ h \neq e}}^{E} p_{he}^{src} \left(\int_0^{\infty} \boldsymbol{\eta}^h e^{\boldsymbol{Q}_h^* t} \alpha_h e^{-\alpha_h t} dt \right) \boldsymbol{R}_{he}, \tag{6}$$

for all stages $e = 1, \ldots, E$, where p_{he}^{src}, given in (5)*, is the probability of a jump to stage e having originated from stage $h \neq e$.*

Running Example. *Assume $N = 2$, $S_2 = 2$. Let $\boldsymbol{\pi}_1 = [\pi(x_1, x_2)]$ and $\boldsymbol{\pi}_2 = [\pi(x_1, x_2, s_{21}, s_{22})]$ be the vectors of equilibrium probabilities for stages 1 and 2, respectively, specified according to the state ordering of the running example. Then for the given parameters*

$$\boldsymbol{\pi}_1 = [0.1780, 0.2289, 0.1486],$$

$$\boldsymbol{\pi}_2 = [0.1221, 0.1423, 0.0474, 0.0921, 0.0405].$$

By basic Markov chain theory, the entry probability vectors are given by

$$\boldsymbol{\eta}_1 = \frac{\boldsymbol{\pi}_2 \boldsymbol{R}_{21}}{\pi_2 \boldsymbol{R}_{21} \mathbf{1}} = [0.2746, 0.4270, 0.2983]$$

$$\boldsymbol{\eta}_2 = \frac{\boldsymbol{\pi}_1 \boldsymbol{R}_{12}}{\pi_1 \boldsymbol{R}_{12} \mathbf{1}} = [0.3204, 0.4120, 0, 0.2676, 0]$$

where $\mathbf{1}$ is a column vector of ones. It can be readily verified that $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ satisfy (6)*, where $p_{12}^{src} = 1$ and $p_{21}^{src} = 1$. For example, using the Laplace transform of the matrix exponential in* (6) *it can be readily seen that $\boldsymbol{\eta}_1 =$*

$\alpha_{21}\boldsymbol{\eta}_2(\alpha_{21}\boldsymbol{I}-\boldsymbol{Q}_2^*)^{-1}\boldsymbol{R}_{21}$ *and* $\boldsymbol{\eta}_2 = \alpha_{12}\boldsymbol{\eta}_1(\alpha_{12}\boldsymbol{I}-\boldsymbol{Q}_1^*)^{-1}\boldsymbol{R}_{12}$*, which can be verified without explicit integration.*

We are now ready to characterise the mean queue-lengths embedded at stage transition instants. Let $\bar{n}_i^h(t)$ be a deterministic function representing the mean number of jobs in station i at time t since entering stage h, conditional on the absence of stage changes occurring during these t time units. Similarly, let $\bar{x}_i^h(t)$ and $\bar{s}_k^h(t)$ be the time averaged values of the state descriptors x_i and s_k in stage h, computed at time t after entering that stage. From (2) it follows that

$$\bar{n}_i^h(t) = \bar{x}_i^h(t) + \sum_{k=2}^{K_{max}} \bar{s}_k^h(t)$$

Denote by $\bar{\boldsymbol{n}}^h(t) = (\bar{n}_1^h(t), \ldots, \bar{n}_M^h(t))$ the corresponding mean queue-length vector. Also, let $\mathcal{L}(\boldsymbol{X}(t);\alpha) = \int_0^\infty \boldsymbol{X}(t)e^{-\alpha t}dt$ be the Laplace transform of vector $\boldsymbol{X}(t)$, $\alpha > 0$.

Corollary 1

$$\bar{\boldsymbol{n}}^e(0) = \sum_{\substack{h=1 \\ h\neq e}}^{E} p_{he}^{src}\alpha_h \boldsymbol{N}^h(\alpha_h), \tag{7}$$

where $\boldsymbol{N}^h(\alpha) = \mathcal{L}(\bar{\boldsymbol{n}}^h(t);\alpha)$.

Proof 1 Using the Kolmogorov forward equations, the state of the system t time units after the environment enters stage h can be written $\boldsymbol{\eta}^h(t) = \boldsymbol{\eta}^h e^{\boldsymbol{Q}_h^* t}$. Since the matrix exponential is a continuous mapping, $\boldsymbol{\eta}^h(t)$ is a continuous vector function by definition. Let $\boldsymbol{v}_i^h$ be the vector of coefficients such that $\bar{n}_i^h(t) = \boldsymbol{\eta}^h(t)\boldsymbol{v}_i^h = \boldsymbol{\eta}^h e^{\boldsymbol{Q}_h^* t}\boldsymbol{v}_i^h$. This vector always exists as it multiplies the probability of a state $(\boldsymbol{n}, h)$ in $\boldsymbol{\eta}^h(t)$ by $n_i^h = x_i^h + \sum_{k=2}^{K_i^e} s_{i,k}$. Using (6), it is then

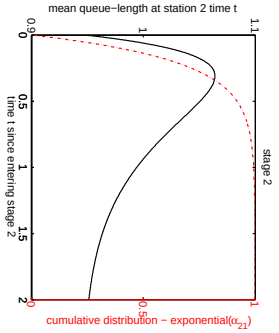
$$\bar{n}_i^e(0) = \boldsymbol{\eta}^e(0)\boldsymbol{v}_i^e = \boldsymbol{\eta}^e\boldsymbol{v}_i^e = \sum_{\substack{h=1 \\ h\neq e}}^{E} p_{he}^{src}\left(\int_0^{+\infty} \boldsymbol{\eta}^h e^{\boldsymbol{Q}_h^* t}\alpha_h e^{-\alpha_h t}dt\right)\boldsymbol{R}_{he}\boldsymbol{v}_i^e,$$

where we used that $\boldsymbol{\eta}^e(0) = \boldsymbol{\eta}^e e^{\boldsymbol{Q}_e^* 0} = \boldsymbol{\eta}^e$. We now observe that we can replace $\boldsymbol{R}_{he}\boldsymbol{v}_i^e$ with $\boldsymbol{v}_i^h$, since the reset rule does not change the total populations at each queue. This implies that

$$\bar{n}_i^e(0) = \sum_{\substack{h=1 \\ h\neq e}}^{E} p_{he}^{src}\alpha_h\left(\int_0^{+\infty} \bar{n}_i^h(t)e^{-\alpha_h t}dt\right).$$

Finally, the Laplace transform exists since $\bar{n}_i^h(t)$ is continuous, which follows by the continuity of $\boldsymbol{\eta}^h(t)$, and always defined for $\alpha_h > 0$, since $\int_0^\infty \boldsymbol{\eta}^h e^{(-\alpha_h\boldsymbol{I}+\boldsymbol{Q}_h^*)t}\boldsymbol{v}_h^e dt = \boldsymbol{\eta}^h(\alpha_h\boldsymbol{I}-\boldsymbol{Q}_h^*)^{-1}\boldsymbol{v}_h^e$ and $(\alpha_h\boldsymbol{I}-\boldsymbol{Q}_h^*)^{-1}$ is always invertible, $\boldsymbol{Q}_h^*$ being a generator. □

Running Example. *From* $\boldsymbol{\eta}_1$*, we can readily compute the mean queue-length at the FCFS queue seen upon stage change as* $\bar{n}_2^1(0) = \boldsymbol{\eta}_1[0,1,2]^T = 1.0237$ *jobs and* $\bar{n}_2^2(0) = \boldsymbol{\eta}_2[0,1,1,2,2]^T = 0.9471$ *jobs. To illustrate* (7)*, we plot the trajectory of* $\overline{\boldsymbol{n}}^2(t)$*:*



The quantity $\alpha_{21}\boldsymbol{N}^2(\alpha_{21})$ (7) *is simply the average of* $\overline{\boldsymbol{n}}^2(t)$ *computed with respect to the stage transition time density* $\alpha_{21}e^{-\alpha_{21}t}$ *and equals* $\overline{\boldsymbol{n}}^1(0) = \alpha_{21}\boldsymbol{N}^2(\alpha_{21}) = 1.0237$.

Expression (7) cannot be exploited directly to solve the model as it requires the unknown mean queue-length transient trajectories $\bar{n}_i(t)$ in order to compute the Laplace transform terms. The blending algorithm attempts to overcome this issue by developing a fixed point iteration with the goal of approximating these trajectories. A key advantage of this approach compared to direct numerical evaluation of the CTMC underlying the queueing network is that the fixed point iteration we propose uses a system of ODEs that grows in size only as $O(MK_{max})$, M being the number of queues and $K_{max} = \max_{i=1..E} K_i$ the maximum number of phases across all service processes, which is independent of the number of jobs in the system.

We begin with the observation that, if such trajectories satisfy an initial value problem defined by an ODE system

$$\frac{d\overline{\boldsymbol{n}}^h(t)}{dt} = \boldsymbol{U}^h(\overline{\boldsymbol{n}}^h(t); \overline{\boldsymbol{n}}^h(0)), \tag{8}$$

for some piece-wise differentiable vector function $\boldsymbol{U}^h(\cdot)$ and initial vector $\overline{\boldsymbol{n}}^h(0)$, then the Laplace transforms are readily determined as

$$\alpha\boldsymbol{N}^h(\alpha) = \overline{\boldsymbol{n}}^h(0) + \mathcal{L}(\boldsymbol{U}^h(\overline{\boldsymbol{n}}^h(t)); \alpha),$$

such that (7) becomes

$$\overline{\boldsymbol{n}}^e(0) = \sum_{\substack{h=1 \\ h\neq e}}^{E} p_{he}^{src}\left(\overline{\boldsymbol{n}}^h(0) + \mathcal{L}(\boldsymbol{U}^h(\overline{\boldsymbol{n}}^h(t)); \alpha_h)\right), \quad e = 1,\ldots,E. \tag{9}$$

where the Laplace transform is now evaluated at α_h. The last equation may be used to develop a fixed-point iteration as follows. Let $\overline{\boldsymbol{n}}_c^e(t)$ be the estimate of the $\overline{\boldsymbol{n}}^e(t)$ vector at iteration $c \geq 1$. Then we can replace (9) with the iteration

$$\overline{\boldsymbol{n}}_{c+1}^e(0) = \sum_{\substack{h=1 \\ h \neq e}}^{E} p_{he}^{src} \left(\overline{\boldsymbol{n}}_c^h(0) + \mathcal{L}(\boldsymbol{U}^h(\overline{\boldsymbol{n}}_c^h(t)); \alpha_h) \right), \tag{10}$$

for all stages $e = 1, \ldots, E$. By iteratively evaluating (10), the blending algorithm aims at finding a fixed point solution $\overline{\boldsymbol{n}}_\infty^e(t)$ satisfying

$$\overline{\boldsymbol{n}}_\infty^e(0) = \sum_{\substack{h=1 \\ h \neq e}}^{E} p_{he}^{src} \left(\overline{\boldsymbol{n}}_\infty^h(0) + \mathcal{L}(\boldsymbol{U}^h(\overline{\boldsymbol{n}}_\infty^h(t)); \alpha_h) \right), \tag{11}$$

such that we can operate the approximation $\overline{\boldsymbol{n}}^e(t) \approx \overline{\boldsymbol{n}}_\infty^e(t)$, for all stages $e = 1, \ldots, E$. As we show later, we can directly use the estimates $\overline{\boldsymbol{n}}_\infty^e(t)$ to calculate performance measures such as mean throughputs, utilisations, and mean response times at equilibrium for the queueing network.

In the next sections, we investigate the fixed point iteration (10), first defining in Section 5 a suitable vector function $\boldsymbol{U}^h(\cdot)$ to approximate the $\overline{n}_i(t)$ trajectories in closed queueing networks, then introducing in Section 6 the blending algorithm. The Laplace transform analysis will be exploited in particular in Section 7 for convergence analysis.

5 Fluid Limits

In this section, we use the theory of fluid limits developed by Kurtz [28] to define an approximate ODE analysis method for queueing network models. This will provide the form of the ODE system (8) to be used in the fixed point iteration (10). The parameters and espressions are taken to refer to an arbitrary stage h; however, in this section we omit this index for ease of readability. The state descriptor of the CTMC now simplifies to $\boldsymbol{Y} = \boldsymbol{n}$.

For queueing networks, the fluid limit we consider describes the behaviour of the Markov process underlying the queueing network as the population of jobs N and the number of servers S_i at each queue i grow simultaneously to infinity. Such growth is assumed to be proportional to a scale parameter V, which increases both the number of servers and jobs in the system, but keeps their ratio constant. For example, when $V = 10$, a model with single server queues and $N = 10$ jobs is scaled to a model with $N' = 100$ jobs and multi-server queues each having $S_i' = 10$ servers. Since capacity grows at the same factor of the population growth, the network avoids saturation

in the asymptotic regime. In this limit, the network also maintains the same topology, routing matrix, and service rates of the unscaled model.

For the class of models we consider, and recalling our assumption of conditioning upon a stage, a fluid limit with the above properties is obtained by taking a family of CTMCs $\{\boldsymbol{Y}_V(t), V \in \mathbb{N}\}$ indexed by the scale parameter V. Each vector $\boldsymbol{Y}_V(t)$ is the state vector for the system at time t, obtained by observing a sample path of the CTMC underlying the queueing network model instantiated with scale parameter V. Let $\boldsymbol{Y}_0$ be the initial state of the CTMC at time $t = 0$, given by the initial placements of jobs across the stations and the initial phase of all servers. Then $\boldsymbol{Y}_1(t)$, where $V = 1$, is the state vector reachable from $\boldsymbol{Y}_0$ through the rates in Table 1, conditional on the environment remaining in the same stage. Note that the case $V = 1$ corresponds to the original model being investigated, thus $\boldsymbol{Y}_1(t) = \boldsymbol{n}(t)$. The vector $\boldsymbol{Y}_2(t)$, where $V = 2$, is the state vector reachable from $2\boldsymbol{Y}_0$ through the same rates. Generalising, $\boldsymbol{Y}_V(t)$ is the state vector reachable from $V\boldsymbol{Y}_0$, for all $V \in \mathbb{N}$. Recalling that each state $\boldsymbol{Y}$ includes both a description of jobs and servers, it follows that using $V\boldsymbol{Y}_0$ as initial state scales both the number of jobs and the number of servers in the queueing network.

The fluid approximation of interest here comprises the limit behaviour of the *normalised* CTMC $\boldsymbol{Y}_V(t)/V$ when V grows asymptotically large, with initial state $\boldsymbol{Y}_0$. A necessary condition to apply the limit theorem is that all transition rates enjoy the so-called *density-dependent* form [28], which requires that all CTMC transition rates need to be proportional to V once rewritten in terms of the normalised state descriptor $\boldsymbol{Y}/V = \boldsymbol{n}/V$. It is easy to verify that all the transition rates in Table 1 can be expressed in such a form – except for f^{env}, which, however, we do not consider here having conditioned the analysis on a given stage. For instance, making explicit the normalised state descriptor $\boldsymbol{Y}/V = \boldsymbol{n}/V$ in the f_1^{dep} rates yields

$$\begin{aligned} f_1^{dep}(\boldsymbol{n}, i, j) &= \phi_{i1} r_{ij} \mu_{i1} \min(x_i, s_{i1}) \\ &= V \phi_{i1} r_{ij} \mu_{i1} \min(x_i/V, s_{i1}/V) \\ &= V f_1^{dep}(\boldsymbol{n}/V, i, j), \end{aligned} \tag{12}$$

which is exactly the density-dependent form. In Kurtz's theory, this form implies that, in the limit, a sample path of the normalised CTMC takes increasingly small steps, of order $O(1/V)$, at increasingly large rates, of order $O(V)$. As $V \to +\infty$, a sample path can therefore be related to a continuous trajectory. Kurtz shows that this trajectory is the solution of a system of ordinary differential equations (ODEs), defined in terms of the density-dependent transition rates of the CTMC. The structure of this system of ODEs is discussed in the next subsection.

Using the methodology developed by Kurtz, the fluid approximation defined above is evaluated as follows. We consider the limit $V \to +\infty$ and approximate the state descriptor $\boldsymbol{Y}_V(t)$ leveraging on a deterministic trajectory $\boldsymbol{y}(t)$ defined by the nonlinear ODE system

$$\frac{d\boldsymbol{y}(t)}{dt} = \sum_{i,j=1}^{M} \sum_{k=1}^{K_i} f_k^{dep}(\boldsymbol{y}(t), i, j)\boldsymbol{\delta}_k^{dep}(i,j) + \sum_{i=1}^{M} \sum_{k=1}^{K_i} f_k^{ph}(\boldsymbol{y}(t), i)\boldsymbol{\delta}_k^{ph}(i), \qquad (13)$$

where $\delta_k^{dep}(i,j)$ and $\boldsymbol{\delta}_k^{ph}(i)$ are *jump vectors* that record the net change of probability mass due to a transition of type $f_k^{dep}(\boldsymbol{y}(t), i, j)$ and $f_k^{ph}(\boldsymbol{y}(t), i)$, respectively. The jump vector for each transition rate function is constructed by taking $\boldsymbol{n}' - \boldsymbol{n}$ in the corresponding row in Table 1. Note that (13) is mass-conserving by the definitions in Table 1.

Let $\boldsymbol{y}(0)$ be the initial state of (13). Using Kurtz's theorem [28, Theorem 3.1] and the fact that the transition rates are Lipschitz continuous everywhere, which implies global existence and uniqueness of the initial value problem (13), it can be shown that, for any finite T and any $\varepsilon > 0$,

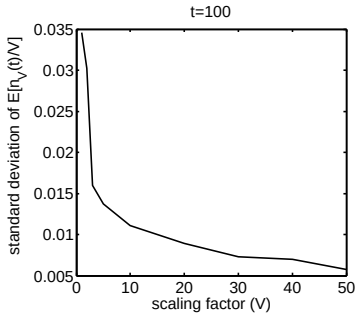
$$\lim_{V\to\infty} \mathbb{P}\left\{\sup_{t\leq T} |\boldsymbol{Y}_V(t)/V - \boldsymbol{y}(t)| > \varepsilon\right\} = 0. \qquad (14)$$

This implies that a sample path (of finite length) is asymptotically indistinguishable, in probability, from the unique solution to the ODE system (13). Furthermore, this fluid limit can be shown to imply *convergence in mean* [6]:

$$\lim_{V\to\infty} \mathbb{E}\left\{|\boldsymbol{Y}_V(t)/V - \boldsymbol{y}(t)|\right\} = 0, \quad t \geq 0, \qquad (15)$$

which provides the motivation for considering the approximation $\mathbb{E}\{\boldsymbol{Y}_V(t)\} \approx V\boldsymbol{y}(t)$ that relates the expected queue length to the re-scaled ODE trajectory. The ODE solution $\boldsymbol{y}(t)$ will be used to approximate the mean performance measures of the original queueing network in which $V = 1$.

Running Example. *We assume $N = V$, $S_2 = V$ and determine the standard deviation of the mean number of jobs at the FCFS queue at time $t = 100$ for increasing values of V, computed over 30 sample paths and conditional on the system being in stage 2. The resulting plot is as follows:*



Consistently with (15)*, the figure shows that as* V *grows the system becomes increasingly deterministic, thus easier to approximate with the trajectory of an ODE system. Notice in particular that even small values of* V *can produce a sharp decrease of variability.*

5.2 Building the ODE System

Despite its apparently complex form, (13) is in practice simple to specify. Let us define $\boldsymbol{R} = [r_{ij}]_{M \times M}$ to be the routing probability matrix of the queueing network model. For example, in a queueing network with exponential servers $\overline{\boldsymbol{n}}(t) = \overline{\boldsymbol{x}}(t)$, where $\overline{\boldsymbol{n}}(t) = (\overline{n}_1(t), \ldots, \overline{n}_M(t))$ and $\overline{\boldsymbol{x}}(t) = (\overline{x}_1(t), \ldots, \overline{x}_M(t))$ being $\overline{x}_i(t)$ the mean value of $x_i(t)$ at time t. Since all servers are in phase 1, the ODE system admits the simplified form

$$\frac{d\overline{\boldsymbol{n}}(t)}{dt} = (-\boldsymbol{I} + \boldsymbol{R}^T)\boldsymbol{\mu}_1 \min(\overline{\boldsymbol{n}}(t), \boldsymbol{S}), \tag{16}$$

where the minimum of two vectors is taken element-wise, the vector $\boldsymbol{S} = (S_1, \ldots, S_M)$ collects the number of servers at each station, $\boldsymbol{\mu}_1 = diag(\mu_{11}, \ldots, \mu_{M1})$ is the vector of service rates at the exponential phase, and the ODE coefficient

matrix is thus

$$(-\boldsymbol{I}+\boldsymbol{R}^T)\boldsymbol{\mu}_1 = \begin{bmatrix} -(1-r_{1,1})\mu_{1,1} & r_{2,1}\mu_{2,1} & \dots & r_{M,1}\mu_{M,1} \\ r_{1,2}\mu_{1,1} & -(1-r_{2,2})\mu_{2,1} & \dots & r_{M,2}\mu_{M,1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1,M}\mu_{1,1} & r_{2,M}\mu_{2,1} & \dots & -(1-r_{M,M})\mu_{M,1} \end{bmatrix}. \quad (17)$$

Notice that the coefficient matrix includes only rates pertaining to the f_1^{dep} transitions in Table 1 since PHs are exponential and that its transpose is an infinitesimal generator since $\sum_{j=1}^{M} r_{i,j} = 1$, for all $i = 1, \dots, M$. [3]

Running Example. *The ODE system* (17) *can describe stage* 1*, where service times are exponentially distributed. Let* $\boldsymbol{S} = (S_1, S_2) = (\infty, 2)$*,* $r_{1,2} = r_{2,1} = 1$*,* $r_{1,1} = r_{2,2} = 0$*,* $\mu_{1,1} = \mu$*,* $\mu_{2,1} = \sigma_1$*. Thus*

$$(-\boldsymbol{I} + \boldsymbol{R}^T)\boldsymbol{\mu}_1 = \begin{bmatrix} -\mu & \sigma_1 \\ \mu & -\sigma_1 \end{bmatrix}$$

and thus $((-\boldsymbol{I}+\boldsymbol{R}^T)\boldsymbol{\mu}_1)^T$ *is an infinitesimal generator. Recall that* $\bar{n}_i^e(t)$ *is the mean queue-length at station* i *at time* t*, when the environment is in stage* e*. Then the ODE system may be written as*

$$\begin{aligned} \frac{d\bar{n}_1^1(t)}{dt} &= -\mu\bar{n}_1^1(t) + \sigma_1 \min(\bar{n}_2^1(t), S_2) \\ \frac{d\bar{n}_2^1(t)}{dt} &= \mu\bar{n}_1^1(t) - \sigma_1 \min(\bar{n}_2^1(t), S_2) \end{aligned}$$

for arbitrary initial conditions $\bar{n}_1^1(0)$*,* $\bar{n}_2^1(0)$*.*

In the general case where service processes are Coxian instead of exponential, the ODE system has the following structure. Let us partition the vector of unknowns as $\boldsymbol{y}(t) = (\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{s}}_1(t), \dots, \bar{\boldsymbol{s}}_{K_{\max}}(t))$, where $K_{\max} = \max_{i=1,\dots,E} K_i$ and $\bar{\boldsymbol{s}}_k(t) = (s_{1k}(t), \dots, s_{Mk}(t))$ for phase k, where we set $s_{ik}(t) \equiv 0$ if $k > K_i$. Compared to the previous case, we have now that servers may be in different phases, but their number does not change within a stage, thus

$$\sum_{k=1}^{K_{\max}} \bar{\boldsymbol{s}}_k(t) = \boldsymbol{S}. \quad (18)$$

[3] Throughout the next sections, we use the ODE system convention of storing state variables in column vectors. By taking the transpose of our equations it is possible to retrieve the row-oriented notation used in Markov process analysis.

Recalling the definitions in Section 3, we now define for phase $k = 1, \ldots, K_{\max}$ the diagonal matrix $\boldsymbol{\mu}_k = diag(\mu_{1k}, \ldots, \mu_{Mk})$ to be the service rate matrix and $\boldsymbol{\Phi}_k = diag(\phi_{1k}, \ldots, \phi_{Mk})$ to be the service completion probability matrix, where we set $\phi_{iK_i} = 1$ and $\mu_{ik} = 0$ if $k > K_i$. The ODE system then becomes

$$\frac{d\bar{\boldsymbol{x}}(t)}{dt} = (-\boldsymbol{I} + \boldsymbol{R}^T\boldsymbol{\Phi}_1)\boldsymbol{\mu}_1 \min(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{s}}_1(t)) + \textstyle\sum_{k=2}^{K_{\max}} \boldsymbol{R}^T\boldsymbol{\Phi}_k\boldsymbol{\mu}_k\bar{\boldsymbol{s}}_k(t), \quad (19)$$

$$\frac{d\bar{\boldsymbol{s}}_2(t)}{dt} = -\boldsymbol{\mu}_2\bar{\boldsymbol{s}}_2(t) + (\boldsymbol{I} - \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1 \min(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{s}}_1(t)), \quad (20)$$

$$\frac{d\bar{\boldsymbol{s}}_k(t)}{dt} = -\boldsymbol{\mu}_k\bar{\boldsymbol{s}}_k(t) + (\boldsymbol{I} - \boldsymbol{\Phi}_{k-1})\boldsymbol{\mu}_{k-1}\bar{\boldsymbol{s}}_{k-1}(t), \qquad k = 3, \ldots, K_{\max} \quad (21)$$

where by (18) it is $\bar{\boldsymbol{s}}_1(t) = \boldsymbol{S} - \sum_{k=2}^{K_{\max}} \bar{\boldsymbol{s}}_k(t)$. The mean queue-lengths are computed by the usual relationship $\bar{\boldsymbol{n}}(t) = \bar{\boldsymbol{x}}(t) + \sum_{k=2}^{K_{max}} \bar{\boldsymbol{s}}_k(t)$.

Running Example. *The ODE system* (19)-(21) *can be used to describe the queueing network while the environment is in stage* 2. *Here* $\boldsymbol{S} = (S_1, S_2) = (\infty, 2)$, $r_{1,2} = r_{2,1} = 1$, $r_{1,1} = r_{2,2} = 0$, $\mu_{1,1} = \mu$, $\mu_{2,1} = \sigma_2$. *Furthermore,*

$$\boldsymbol{R} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \boldsymbol{\Phi}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{\Phi}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{\mu}_1 = \begin{bmatrix} \mu & 0 \\ 0 & \sigma_2 \end{bmatrix}, \boldsymbol{\mu}_2 = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_2 \end{bmatrix},$$

thus (19)-(21) *become*

$$\frac{d\bar{x}_1(t)}{dt} = -\mu\bar{x}_1(t) + \sigma_2\bar{s}_{2,2}(t) \quad (22)$$

$$\frac{d\bar{x}_2(t)}{dt} = \mu\bar{x}_1(t) - \sigma_2 \min(\bar{x}_2(t), \bar{s}_{2,1}(t)) \quad (23)$$

$$\frac{d\bar{s}_{2,2}(t)}{dt} = -\sigma_2\bar{s}_{2,2}(t) + \sigma_2 \min(\bar{x}_2(t), \bar{s}_{2,1}(t)) \quad (24)$$

where $\bar{s}_{2,1}(t) = S_2 - \bar{s}_{2,2}(t)$.

The ODE system (19)-(21) can be solved directly by a numerical nonlinear ODE solver. The initial values are $\bar{\boldsymbol{s}}_2(0) = \ldots = \bar{\boldsymbol{s}}_{K_{\max}}(0) = \boldsymbol{0}$, to impose that jobs start service from phase 1, and a queue-length vector $\bar{\boldsymbol{x}}(0) \geq \boldsymbol{0}$, $\sum_j \bar{\boldsymbol{x}}_j(0) = N$, such that the total population of jobs is N. Since the ODE system size is $M + \sum_{i=1}^{M}(K_i^e - 1)$ equations and variables, the fluid approximation provides major computational savings compared to a CTMC analysis where the number of global balance equations and variables grows exponentially as $O(EN^M \prod_{i=1}^{K_i} S_i^{K_i})$. As explained before, despite its small size, the fluid approximation provides accuracy and convergence guarantees as the system size increases.

We now seek the expression for the Laplace transform used in Corollary 7. From this, we develop in Section 6 the blending algorithm, using the iterative approach outlined in Section 4.

Thanks to Lipschitz continuity, the ODE system (19)-(21) has a unique solution with piece-wise continuous derivatives. Since the network is closed, both vectors $\bar{\boldsymbol{x}}(t)$ and $\bar{\boldsymbol{s}}_k(t)$, $k \geq 2$, have to remain bounded, since $\bar{s}_k^h(t) \leq \min(N, S_k)$ and $\bar{x}_k(t) \leq N$ for all queues k and stages h, thus their Laplace transform exists together with their derivatives. This makes the fluid approach suitable for use within the iterative analysis outlined in Section 4. Let

$$\boldsymbol{X}(\alpha) = \mathcal{L}(\bar{\boldsymbol{x}}(t);\alpha), \quad \boldsymbol{M}(\alpha) = \mathcal{L}(\min(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{s}}_1(t));\alpha), \quad \boldsymbol{S}_k(\alpha) = \mathcal{L}(\bar{\boldsymbol{s}}_k(t);\alpha).$$

The Laplace transform of the ODE system is

$$\alpha\boldsymbol{X}(\alpha) = \bar{\boldsymbol{x}}(0) + (-\boldsymbol{I} + \boldsymbol{R}^T\boldsymbol{\Phi}_1)\boldsymbol{\mu}_1\boldsymbol{M}(\alpha) + \textstyle\sum_{k=2}^{K_{\max}} \boldsymbol{R}^T\boldsymbol{\Phi}_k\boldsymbol{\mu}_k\boldsymbol{S}_k(\alpha), \tag{25}$$

$$\alpha\boldsymbol{S}_2(\alpha) = -\boldsymbol{\mu}_2\boldsymbol{S}_2(\alpha) + (\boldsymbol{I} - \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1\boldsymbol{M}(\alpha), \tag{26}$$

$$\alpha\boldsymbol{S}_k(\alpha) = -\boldsymbol{\mu}_k\boldsymbol{S}_k(\alpha) + (\boldsymbol{I} - \boldsymbol{\Phi}_{k-1})\boldsymbol{\mu}_{k-1}\boldsymbol{S}_{k-1}(\alpha), \qquad k = 3, \ldots, K_{\max} \tag{27}$$

where $\bar{\boldsymbol{s}}_k(0) = \boldsymbol{0}$, $k \geq 2$ by the reset rule. Bringing $\boldsymbol{S}_2(\alpha)$ and $\boldsymbol{S}_k(\alpha)$ to the left hand side and noting that $\alpha\boldsymbol{I} + \boldsymbol{\mu}_k$, $k \geq 2$, is a diagonal matrix with strictly positive entries, and so always invertible, (26)-(27) imply

$$\boldsymbol{S}_k(\alpha) = \prod_{j=k\ldots 2} \left((\alpha\boldsymbol{I} + \boldsymbol{\mu}_j)^{-1}(\boldsymbol{I} - \boldsymbol{\Phi}_{j-1})\boldsymbol{\mu}_{j-1}\right) \boldsymbol{M}(\alpha), \qquad k = 2, \ldots, K_{\max}$$

Since the Laplace transform of the queue-length vector is $\boldsymbol{N}(\alpha) = \mathcal{L}(\bar{\boldsymbol{n}}(t);\alpha) = \boldsymbol{X}(\alpha) + \sum_{k=2}^{K_{\max}} \boldsymbol{S}_k(\alpha)$, where $n_j = x_j + \sum_{k=2}^{K_j} s_{j,k}$ and (by convention) $\boldsymbol{S}_k = \boldsymbol{0}$ for $k > K_i$, we conclude that

$$\alpha\boldsymbol{N}(\alpha) = \bar{\boldsymbol{n}}(0) + \boldsymbol{B}(\alpha)\boldsymbol{M}(\alpha) = \bar{\boldsymbol{x}}(0) + \boldsymbol{B}(\alpha)\boldsymbol{M}(\alpha), \tag{28}$$

where the matrix

$$\boldsymbol{B}(\alpha) = (-\boldsymbol{I} + \boldsymbol{R}^T\boldsymbol{\Phi}_1)\boldsymbol{\mu}_1 + \sum_{k=2}^{K_{\max}} (\boldsymbol{R}^T\boldsymbol{\Phi}_k\boldsymbol{\mu}_k + \alpha\boldsymbol{I}) \prod_{j=k\ldots 2} (\boldsymbol{\mu}_j + \alpha\boldsymbol{I})^{-1}(\boldsymbol{I} - \boldsymbol{\Phi}_{j-1})\boldsymbol{\mu}_{j-1} \tag{29}$$

depends only on the input parameters of the model and has the following property.

Theorem 1 *The matrix $(\boldsymbol{B}(\alpha))^T$ is an infinitesimal generator.*

Proof 2 *Let* $\mathbf{1}^T$ *be a row vector of ones. We begin by proving that* $(\boldsymbol{R}^T\boldsymbol{\Phi}_{K_{max}}\boldsymbol{\mu}_{K_{max}}+\alpha\boldsymbol{I})(\alpha\boldsymbol{I}+\boldsymbol{\mu}_{K_{max}})^{-1}$ *is a column-stochastic matrix. We first note that the entries of this matrix are non-negative since all its constituent matrices are non-negative and the inverse matrix is diagonal. Thus, to prove stochasticity, we just need to show that the columns of the matrix sum to unity. Recall that* $\boldsymbol{\Phi}_{K_{max}}$ *is a binary matrix, since all Coxian service processes must either complete at an earlier phase, in which case* $\phi_{i,K_{max}} = 0$ *and* $\mu_{i,k} = 0$*, or complete at phase* K_{max} *implying* $\phi_{i,K_{max}} = 1$ *and* $\mu_{i,k} > 0$*. This implies that* $\boldsymbol{\Phi}_{K_{max}}\boldsymbol{\mu}_{K_{max}} = \boldsymbol{\mu}_{K_{max}}$*. Therefore*

$$\mathbf{1}^T(\boldsymbol{R}^T\boldsymbol{\Phi}_{K_{max}}\boldsymbol{\mu}_{K_{max}}+\alpha\boldsymbol{I})(\boldsymbol{\mu}_{K_{max}}+\alpha\boldsymbol{I})^{-1} = (\mathbf{1}^T\boldsymbol{\mu}_{K_{max}}+\alpha\mathbf{1}^T)(\boldsymbol{\mu}_{K_{max}}+\alpha\boldsymbol{I})^{-1} = \mathbf{1}^T$$

where we used the facts that $\mathbf{1}^T\boldsymbol{R}^T = \mathbf{1}^T$ *and* $\boldsymbol{\mu}_{K_{max}}$ *is diagonal.*

We now focus on the summation in (29)*. Using the matrix recurrence*

$$\boldsymbol{F}(j) = (\boldsymbol{F}(j+1) + (\boldsymbol{R}^T\boldsymbol{\Phi}_j\boldsymbol{\mu}_j + \alpha\boldsymbol{I}))(\boldsymbol{\mu}_j + \alpha\boldsymbol{I})^{-1}(\boldsymbol{I} - \boldsymbol{\Phi}_{j-1})\boldsymbol{\mu}_{j-1}$$

with $\boldsymbol{F}(K_{max}+1) = \mathbf{0}$*, the summation in* (29) *can be computed as* $\boldsymbol{F}(1)$*. From the above definition, we then have*

$$\begin{aligned}\mathbf{1}^T\boldsymbol{F}(K_{max}) =& \mathbf{1}^T(\boldsymbol{R}^T\boldsymbol{\Phi}_{K_{max}}\boldsymbol{\mu}_{K_{max}} + \alpha\boldsymbol{I}))(\boldsymbol{\mu}_{K_{max}} + \alpha\boldsymbol{I})^{-1}(\boldsymbol{I} - \boldsymbol{\Phi}_{K_{max}-1})\boldsymbol{\mu}_{K_{max}-1}\\ =& \mathbf{1}^T(\boldsymbol{I} - \boldsymbol{\Phi}_{K_{max}-1})\boldsymbol{\mu}_{K_{max}-1}\end{aligned}$$

which allows us to determine the next value in the recurrence as

$$\begin{aligned}\mathbf{1}^T\boldsymbol{F}(K_{max}-1) =& \mathbf{1}^T((\boldsymbol{I} - \boldsymbol{\Phi}_{K_{max}-1})\boldsymbol{\mu}_{K_{max}-1} + (\boldsymbol{R}^T\boldsymbol{\Phi}_{K_{max}-1}\boldsymbol{\mu}_{K_{max}-1} + \alpha\boldsymbol{I}))\times\\ &\times (\boldsymbol{\mu}_{K_{max}-1} + \alpha\boldsymbol{I})^{-1}(\boldsymbol{I} - \boldsymbol{\Phi}_{K_{max}-2})\boldsymbol{\mu}_{K_{max}-2}\\ =& \mathbf{1}^T(\boldsymbol{I} - \boldsymbol{\Phi}_{K_{max}-2})\boldsymbol{\mu}_{K_{max}-2}\end{aligned}$$

and thus recursively we obtain $\mathbf{1}^T\boldsymbol{F}(1) = \mathbf{1}^T(\boldsymbol{I} - \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1$*. This implies that*

$$\begin{aligned}\mathbf{1}^T\boldsymbol{B}(\alpha) =& \mathbf{1}^T(-\boldsymbol{I} + \boldsymbol{R}^T\boldsymbol{\Phi}_1)\boldsymbol{\mu}_1 + \mathbf{1}^T(\boldsymbol{I} - \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1\\ =& -\mathbf{1}^T\boldsymbol{\mu}_1 + (\mathbf{1}^T\boldsymbol{R}^T\boldsymbol{\Phi}_1 + (\boldsymbol{I} - \boldsymbol{\Phi}_1))\boldsymbol{\mu}_1\\ =& -\mathbf{1}^T\boldsymbol{\mu}_1 + \mathbf{1}^T\boldsymbol{\mu}_1 = \mathbf{0}.\end{aligned}$$

Thus $\boldsymbol{B}(\alpha)$ *has columns that sum to zero. The result follows by noting that the only negative term in the definition of* $\boldsymbol{B}(\alpha)$ *is the diagonal matrix* $-\boldsymbol{\mu}_1$ *whereas all off-diagonal entries are non-negative by the structure of* (29)*, in which off-diagonal elements are defined by products of non-negative matrices.* □

Note that we cannot further simplify (28), since $\boldsymbol{M}(\alpha)$ is the Laplace transform of $\min(\bar{\boldsymbol{x}}(t), \bar{\boldsymbol{s}}_1(t))$, which cannot be derived analytically unless an explicit expression for the trajectories $\bar{\boldsymbol{x}}(t)$ and $\bar{\boldsymbol{s}}_1(t)$ is known. However, we cannot obtain an explicit solution for ODE systems including minimum functions

except in special cases. This issue is common in other problem areas as well; for example, process algebras that use Kurtz's fluid approximation all rely on numerical ODE solutions due to the presence of minimum functions, e.g. PEPA [41], GPEPA [39].

Running Example. *We wish to illustrate* (28) *under increasing scaling factors* V*. Assume first* $V = 1$*,* $N = 2V = 2$*,* $S_2 = V = 1$*. Using the results of the Running Example in Section 4, we have* $\overline{\boldsymbol{n}}^2(0) = [1.0529, 0.9471]$*. Let* $\boldsymbol{M}^2(\alpha)$ *and* $\boldsymbol{B}^2(\alpha)$ *be* $\boldsymbol{M}(\alpha)$ *and* $\boldsymbol{B}(\alpha)$ *conditional on the system being in stage* 2*. Then we can determine* $\boldsymbol{M}^2(\alpha_2)$ *by the definition after evaluating* (22)-(24) *with initial condition* $\overline{\boldsymbol{n}}^2(0)$ *and* $s_{2,2}(0) = 0$*. We evaluate the ODE system using MATLAB's* `ode15s` *with numerical tolerance set to* 10^{-12} *and find*

$$\boldsymbol{M}^2(\alpha_{21}) = \begin{bmatrix} 0.2023 \\ 0.1430 \end{bmatrix}, \boldsymbol{B}^2(\alpha_{21}) = \begin{bmatrix} -1.0000 & 1.1250 \\ 1.0000 & -1.1250 \end{bmatrix}, \boldsymbol{B}^2(\alpha_{21})\boldsymbol{M}^2(\alpha_{21}) = \begin{bmatrix} -0.0414 \\ 0.0414 \end{bmatrix}$$

which implies $\overline{\boldsymbol{n}}^2(0) + \boldsymbol{B}^2(\alpha_2)\boldsymbol{M}^2(\alpha_2) = [1.0115, 0.9885]$*. Using the ODE system solution we also find* $\alpha_2\boldsymbol{N}^2(\alpha_2) = [1.0113, 0.9887]$*, which equals* $\overline{\boldsymbol{n}}^2(0) + \boldsymbol{B}^2(\alpha_2)\boldsymbol{M}^2(\alpha_2)$ *except for minor numerical errors. This illustrates that* (28) *holds, but also that the ODE system is approximate, since* $\alpha_2\boldsymbol{N}^2(\alpha_2) \neq \overline{\boldsymbol{n}}^1(0) = [0.9763, 1.0237]$*, where the latter is the exact value computed from the CTMC.*

Assume now $V = 10$*,* $N = 2V = 20$*,* $S_2 = V = 10$*. Under this parameterisation the CTMC solution gives* $\overline{\boldsymbol{n}}^2(0)/V = [1.1978, 0.8022]$*. Solving the ODE system with this initial point and* $s_{2,2}(0) = 0$*, we find* $\alpha_2\boldsymbol{N}^2(\alpha_2)/V = [1.1189, 0.8811]$*,* $\overline{\boldsymbol{n}}^2(0)/V + \boldsymbol{B}^2(\alpha_2)\boldsymbol{M}^2(\alpha_2)/V = [1.1185, 0.8815]$*. The exact CTMC value is* $\overline{\boldsymbol{n}}^1(0)/V = [1.1120, 0.8880]$ *showing, compared to the previous case, a progressive convergence of* $\alpha_2\boldsymbol{N}^2(\alpha_2)/V$ *to* $\overline{\boldsymbol{n}}^2(0)/V$ *as* V *grows.*

6 Blending Algorithm

6.1 Blending Iteration

Recall that (28) has been obtained subject to the ODE evolving in a given stage h. This dependence can now be reinstated in the notation as

$$\alpha\boldsymbol{N}^h(\alpha) = \overline{\boldsymbol{n}}^h(0) + \boldsymbol{B}^h(\alpha)\boldsymbol{M}^h(\alpha), \tag{30}$$

where all the matrices defining $\boldsymbol{B}^h(\alpha)$ are instantiated with the stage h model parameters and indicated with $\boldsymbol{R}^h$, $\boldsymbol{\Psi}^h_k$, $\boldsymbol{\mu}^h_k$, $k = 1, \ldots, K_{max}$, $h = 1, \ldots, E$.

The above expression holds under the assumption that the time variable t in the Laplace transforms now represents the time since entering stage h. Under this assumption, (30) uses the fact that at time $t = 0$ all jobs are in phase 1 due to the reset rule, thus $\bar{\boldsymbol{s}}_{k,c}^h = \mathbf{0}$, $k \geq 2$, and we can use the term $\overline{\boldsymbol{n}}_c^h(0)$ since $\overline{\boldsymbol{n}}_c^h(0) = \overline{\boldsymbol{x}}_c^h(0)$.

Following the line of reasoning used in Section 4, we can formulate two equivalent expressions for the fixed point iteration (10). The first expression follows by combining (30) with Corollary 1 and gives the blending algorithm iteration needed to update the mean queue-lengths in each stage, as observed immediately after a stage change:

$$\overline{\boldsymbol{n}}_{c+1}^e(0) = \sum_{\substack{h=1 \\ h \neq e}}^{E} p_{he}^{src} \left(\overline{\boldsymbol{n}}_c^h(0) + \boldsymbol{B}^h(\alpha_h) \boldsymbol{M}_c^h(\alpha_h; \overline{\boldsymbol{n}}_c^E(0)) \right), \tag{31}$$

for $e = 1, \ldots, E$, where $\boldsymbol{M}_c^h(\alpha_h; \overline{\boldsymbol{n}}_c^h(0)) = \mathcal{L}\left(\min(\overline{\boldsymbol{x}}_c^h(t), \boldsymbol{S} - \sum_{k=2}^{K_{\max}^h} \bar{\boldsymbol{s}}_{k,c}^h(t)), \alpha_h \right)$, in which we make explicit the dependence of the transform on the initial vector $\overline{\boldsymbol{n}}_c^h(0)$ used in the ODE system for stage h at iteration c.

Next, we develop an expression that consolidates all equations (31) into a single matrix formula. This second expression is useful for the convergence analysis developed in Section 7. Let $\overline{\boldsymbol{n}}_c(0) = (\overline{\boldsymbol{n}}_c^1(0), \ldots, \overline{\boldsymbol{n}}_c^E(0))$, for $c \geq 1$, be the vector of mean queue-lengths conditional on the current stage, such that $||\overline{\boldsymbol{n}}_c(0)||_1 = NE$. Also, let $\boldsymbol{P} = [p_{he}^{src} \boldsymbol{I}_{M \times M}]_{e,h=1,\ldots,E}$, which is a row-stochastic matrix of order $E^2 M^2$ with zero diagonal. [4]. Expression (31) can now be rewritten as

$$\overline{\boldsymbol{n}}_{c+1}(0) = \boldsymbol{P} \left(\overline{\boldsymbol{n}}_c(0) + \boldsymbol{B} \boldsymbol{M}_c(\overline{\boldsymbol{n}}_c(0)) \right), \tag{32}$$

for all iterations $c \geq 1$, where

$$\boldsymbol{B} = diag(\boldsymbol{B}^1(\alpha_1), \ldots, \boldsymbol{B}^E(\alpha_E)) \tag{33}$$

$$\boldsymbol{M}_c(\overline{\boldsymbol{n}}_c(0)) = [\boldsymbol{M}_c^1(\alpha_1; \overline{\boldsymbol{n}}_c^1(0)); \ldots; \boldsymbol{M}_c^E(\alpha_E; \overline{\boldsymbol{n}}_c^E(0))] \tag{34}$$

$\boldsymbol{B}$ is block diagonal and $\boldsymbol{M}_c(\overline{\boldsymbol{n}}_c(0))$ is a column vector obtained by concatenation. Equations (31) and (32) are equivalent and define the fundamental equations of the fixed point iteration used by the blending algorithm to approximate queueing network models in random environments. An algorithmic description together with details on initial and termination conditions and implementation guidelines are provided in the next section.

[4] Recall that the probabilities p_{eh}^{src} are the probabilities of jumping to stage h after leaving stage e. Since we assume that the stage changes, it must be that $p_{ee}^{src} = 0$, $\forall h$.

Running Example. *Assume* $V = 1$, $N = 2V = 2$, $S_2 = V = 1$. *We initialise the iteration using the exact values determined by the CTMC, i.e.,* $\overline{\boldsymbol{n}}_1(0) = [0.9763, 1.0237, 1.0529, 0.9471]^T$. *Using the definitions we have*

$$\boldsymbol{P} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} -1 & 2 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 0 & 0 & -1 & 1.1250 \\ 0 & 0 & 1 & -1.1250 \end{bmatrix}, \boldsymbol{M}_1(\overline{\boldsymbol{n}}_1(0)) = \begin{bmatrix} 0.2904 \\ 0.2096 \\ 0.2238 \\ 0.1285 \end{bmatrix}$$

Thus $\overline{\boldsymbol{n}}_2(0) = \boldsymbol{P}\left(\overline{\boldsymbol{n}}_1(0) + \boldsymbol{B}\boldsymbol{M}_1(\overline{\boldsymbol{n}}_1(0))\right) = [1.0114, 0.9886, 1.1503, 0.8497]$. *Reapplying the blending iteration* (32) *until* $c = 20$ *we find*

$$\overline{\boldsymbol{n}}_{20}(0) = [1.1354, 0.8646, 1.2208, 0.7792].$$

If we chose a different initial condition, e.g., initialising all jobs at the delay server such that $\overline{\boldsymbol{n}}_1(0) = [2, 0, 2, 0]^T$, *then at iteration* 20 *it is* $\overline{\boldsymbol{n}}_{20}(0) = [1.1357, 0.8643, 1.2210, 0.7790]$ *and at iteration 30 it is*

$$\overline{\boldsymbol{n}}_{30}(0) = [1.1354, 0.8646, 1.2208, 0.7792].$$

This illustrates that the iteration reaches the same fixed point regardless of the initial condition, but the latter affects the rate of convergence. The maximum absolute error of the final estimate compared to the exact value is 17.72%.

6.2 Algorithm and pseudo-code

We now combine the results of the previous sections to formalise the blending algorithm and provide implementation guidelines. Pseudo-code to clarify the implementation of the method is given in Algorithm 1.

The blending algorithm consists of iteratively evaluating (32) until a user-specified number of iterations C is reached or until convergence of the estimated queue-lengths vectors $\overline{\boldsymbol{n}}_c(0)$ is achieved within a specified tolerance, e.g., $||\overline{\boldsymbol{n}}_{c+1}(0) - \overline{\boldsymbol{n}}_c(0)||_1 \leq \delta_{max}$ for some $c \leq C$. Iteration (32) requires the computation of the $\boldsymbol{M}_c^h$ matrices by approximating the state trajectories $\overline{\boldsymbol{x}}^h(t)$ and $\bar{\boldsymbol{s}}_k^h(t)$, $k = 1, \ldots, K_{max}$, $h = 1, \ldots, E$, for all instants $t \in [0, T]$, where T is a user specified maximum time. Such time should be chosen such that the probability of a stage jump is negligible in any stage, e.g., $T = \max\{t : \max_{h,e} e^{\alpha_{he} t} \leq \epsilon\}$, where ϵ is a numerical tolerance (e.g., $\epsilon = 10^{-8}$). At each iteration, the knowledge of the $\overline{\boldsymbol{x}}^h(t)$ and $\bar{\boldsymbol{s}}_k^h(t)$ trajectories gives the ability of computing the mean queue-length trajectories by the usual expression $\boldsymbol{n}_c^h(t) = \overline{\boldsymbol{x}}_c^h(t) + \sum_{k=2}^{K_{max}} \bar{\boldsymbol{s}}_{k,c}^h(t)$, $0 \leq t \leq T$, for all iterations $c \geq 1$.

The blending algorithm aims at finding a fixed point for the trajectories $\overline{\boldsymbol{n}}_c^h(t)$, $\overline{\boldsymbol{x}}_c^h(t)$ and $\bar{\boldsymbol{s}}_{c,k}^h(t)$ over a set of $c \leq C$ iteration, where C is the maximum number of iterations allowed by the user. We have empirically noted that a fixed point exists in most instances, but a formal convergence analysis is carried out in Section 7 to better characterise the algorithm. From this fixed point, which we indicate with the notation $(\overline{\boldsymbol{n}}_\infty^h(t), \overline{\boldsymbol{x}}_\infty^h(t), \bar{\boldsymbol{s}}_{\infty,k}^h(t))$, performance measures can easily be estimated using the expressions we develop in Section 6.3.

An important implementation decision for the blending algorithm is the choice of the initial queue-lengths estimates for the M queues in each stage $h = 1, \ldots, E$. Due to the reset rule, $\boldsymbol{s}_{j,1}(0) = \boldsymbol{S}^h$, $\boldsymbol{s}_{j,k}(0) = \boldsymbol{0}$, $k = 2, \ldots, K_{max}$, and thus $\boldsymbol{n}^h(0) = \boldsymbol{x}^h(0)$ implies that it is sufficient to assign values to the $\boldsymbol{x}^h(0)$ variables for initialisation of the ODEs. Let $\boldsymbol{x}^h(0) = \boldsymbol{x}_0^h$, then $\boldsymbol{x}_0^h$ may be chosen according to an heuristic, for example by placing all jobs at a certain station or by equally diving them across all queues, but preference should be given to distributions that are expected to be close to the equilibrium distribution. For example, decomposition or average-environment based on approximate mean value analysis may be used as alternatives for initialisation, albeit at a price of complicating the implementation.

6.3 Mean Performance Measures

Assume that an exact solution for the model is available. Then the mean queue-length at station j at equilibrium, denoted by Q_j, is readily computed as

$$Q_j = \sum_{h=1}^{E} \pi_h \lim_{T\to\infty} \frac{1}{T} \int_0^T \bar{n}_j^h(t)dt = \sum_{h=1}^{E} \pi_h \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\bar{x}_j^h(t) + \sum_{k=2}^{K_j^h} \bar{s}_{jk}^h(t) \right) dt, \tag{35}$$

which is simply the time average of $\bar{n}_j^h(t)$ in each stage, scaled by the stationary probabilities π_h of that stage. The mean throughput at station j, denoted by X_j, is obtained as

$$X_j = \sum_{h=1}^{E} \pi_h \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\phi_{j,1}^h \mu_{j,1}^h \min(\bar{x}_j^h(t), \bar{s}_{j1}^h(t)) + \sum_{k=2}^{K_j^h} \phi_{j,k}^h \mu_{j,k}^h \bar{s}_{j,k}^h(t) \right) dt, \tag{36}$$

which weights by the probabilities π_h the average departure rates from each Coxian server. Finally, the mean response time W_j per visit at station j may be obtained by Little's law as

$$W_j = \frac{Q_j}{X_j} \tag{37}$$

In order to compute the above measures, the blending algorithm can either obtain a fixed point $(\overline{\boldsymbol{n}}^h_\infty(t), \overline{\boldsymbol{x}}^h_\infty(t), \bar{\boldsymbol{s}}^h_{\infty,k}(t))$ or return a vector $(\overline{\boldsymbol{n}}^h_C(t), \overline{\boldsymbol{x}}^h_C(t), \bar{\boldsymbol{s}}^h_{C,k}(t))$ in case the maximum number of iterations is reached before convergence. The entries of these vectors can be used to estimate (35) and (36) directly by replacing $\bar{x}^h_j(t)$ and $\bar{s}^h_{j,k}(t)$ with the corresponding estimates returned by the algorithm. In particular, the limits in the espressions can be estimated by time-averaging the occupancy measures in the station state vector up to a large enough T, i.e., the T_{max} parameter provided as input to Algorithm 1. Other measures of interest can easily be computed if they can be expressed as a function of the state trajectory.

An issue may arise in the case where the solution of the model does not converge to a fixed point, but oscillates with some period p in the mean queue-length vectors $\overline{\boldsymbol{n}}^h_c(0)$ throughout the iterations. These cases may arise as a result of the fluid approximation; for example we describe in Section 7 a degenerate case of a balanced network showing this behaviour in heavy-load. For such models, the last p vectors $\overline{\boldsymbol{n}}^h_c(0)$ should be averaged to obtain the stationary value of the queue lengths seen at stage transition instants. Such pathological cases may be detected by heuristic tests, e.g., the MSER-5 rule [26].

7 Convergence Analysis

Evaluating the convergence of (32) is challenging due to the intractability of the Laplace transform of the minimum function that appears in the definition of $\boldsymbol{M}_c(\overline{\boldsymbol{n}}_c(0))$. Bounds on the transforms in the expressions are possible, e.g., $\mathcal{L}(\min(\overline{\boldsymbol{x}}(t), \boldsymbol{S}); \alpha) \leq \min(\boldsymbol{X}(\alpha), \alpha^{-1}\boldsymbol{S})$, but in our experience they yield after a few iterations trivial values because they violate the conservation of the fluid mass of the ODE system.

To cope with this intractability, we focus on two special cases where (32) can be evaluated analytically, namely light-load and heavy-load regimes. The following subsections provide detailed convergence analysis results, but first we summarise the main qualitative findings. The light-load and heavy-load analyses suggest that the blending algorithm provides stronger convergence guarantees in light-load, although the heavy-load case is only partly tractable. Our analysis suggests that the algorithm features in both regimes a structure similar to the power method iteration. The power method is an eigenvalue algorithm applied in various domains, including Markov chain analysis, and it is known to converge in most cases except for a few pathological ones, e.g., cyclic matrices. It is possible to construct toy examples to show that such cases can arise also in the application of the blending algorithm, but these appear contrived. Indeed, the numerical experiments reported in Section 8

suggest that the algorithm is quite accurate on average. For pathological cases where the solution oscillates, averaging performance measures over the last few iterations provides a simple way to cope with the lack of a fixed point.

7.1 Convergence analysis in light-load

The light-load convergence analysis assumes that in each stage h the number of jobs at each queue does not exceed its available servers at any point in time. This is a good approximation for a lightly-loaded system where all stations have no significant backlog of waiting jobs and therefore there is a negligible probability that all servers in the queue will become busy. Formally, the assumption is given by

$$\mathcal{L}(\min(\overline{\boldsymbol{n}}_c^h(t), \boldsymbol{S}); \alpha_h) = \mathcal{L}(\overline{\boldsymbol{n}}_c^h(t); \alpha_h) \tag{38}$$

where it should be noted that the minimum in the Laplace transform differs from the one used in the ODE systems discussed in the previous sections, which consider $\bar{\boldsymbol{x}}_c^h(t)$ instead of $\overline{\boldsymbol{n}}_c^h(t)$. However, observing that

$$\mathcal{L}(\overline{\boldsymbol{n}}_c^h(t); \alpha_h) = \boldsymbol{X}_{c,k}^h(\alpha_h) + \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{S}_{c,k}^h(\alpha_h)$$

where $\boldsymbol{X}_c^h(\alpha) = \mathcal{L}(\bar{\boldsymbol{x}}_c^h(t); \alpha)$, and that

$$\begin{aligned}
\mathcal{L}(\min(\overline{\boldsymbol{n}}_c^h(t), \boldsymbol{S}); \alpha_h) =& \mathcal{L}(\min(\overline{\boldsymbol{n}}_c^h(t) - \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{s}_{c,k}^h(t), \boldsymbol{S} - \sum_{k=2}^{K_{max}} \boldsymbol{s}_{c,k}^h(t)); \alpha_h) + \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{S}_{c,k}^h(\alpha_h) \\
=& \mathcal{L}(\min(\bar{\boldsymbol{x}}_c^h(t), \boldsymbol{S} - \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{s}_{c,k}^h(t)); \alpha_h) + \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{S}_{c,k}^h(\alpha_h) \\
=& \boldsymbol{M}_{c,k}^h(\alpha_h) + \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{S}_{c,k}^h(\alpha_h)
\end{aligned}$$

we see that the light-load assumption implies $\boldsymbol{M}_{c,k}^h(\alpha_h) = \boldsymbol{X}_{c,k}^h(\alpha_h)$. Inserting the expression in (32) and simplifying, the blending iteration becomes

$$\overline{\boldsymbol{n}}_{c+1}(0) = \boldsymbol{P}\left(\overline{\boldsymbol{n}}_c(0) + \boldsymbol{B}\boldsymbol{X}_c\right), \tag{39}$$

where $\boldsymbol{X}_c = diag(\boldsymbol{X}_c^1(\alpha_1), \ldots, \boldsymbol{X}_c^E(\alpha_E))$. We can further simplify the expressions as follows. Using $\boldsymbol{M}_{c,k}^h(\alpha_h) = \boldsymbol{X}_{c,k}^h(\alpha_h)$, it follows that (25)-(27) become a system of linear ODEs with transforms satisfying

$$\alpha_h \begin{bmatrix} \boldsymbol{X}^h(\alpha_h) \\ \boldsymbol{S}_2^h(\alpha_h) \\ \vdots \\ \boldsymbol{S}_{K_{\max}}^h(\alpha_h) \end{bmatrix} = \boldsymbol{V}^h \begin{bmatrix} \boldsymbol{X}^h(\alpha_h) \\ \boldsymbol{S}_2^h(\alpha_h) \\ \vdots \\ \boldsymbol{S}_{K_{\max}}^h(\alpha_h) \end{bmatrix} + \begin{bmatrix} \bar{\boldsymbol{x}}^h(0) \\ \bar{\boldsymbol{s}}_2^h(0) \\ \vdots \\ \bar{\boldsymbol{s}}_{K_{\max}}^h(0) \end{bmatrix}.$$

where

$$\boldsymbol{V}^h = \begin{bmatrix} (-\boldsymbol{I} - \boldsymbol{R}^T\boldsymbol{\Phi}_1)\boldsymbol{\mu}_1^h & \boldsymbol{R}_h^T\boldsymbol{\Phi}_2^h\boldsymbol{\mu}_2^h & \boldsymbol{R}_h^T\boldsymbol{\Phi}_3^h\boldsymbol{\mu}_3^h & \cdots & \boldsymbol{R}_h^T\boldsymbol{\mu}_{K_{\max}^h}^h \\ (\boldsymbol{I} - \boldsymbol{\Phi}_1^h)\boldsymbol{\mu}_1^h & -\boldsymbol{\mu}_2^h & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & (\boldsymbol{I} - \boldsymbol{\Phi}_2^h)\boldsymbol{\mu}_2^h & -\boldsymbol{\mu}_3^h & \ddots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & (\boldsymbol{I} - \boldsymbol{\Phi}_{K_{\max}-1}^h)\boldsymbol{\mu}_{K_{\max}-1}^h & -\boldsymbol{\mu}_{K_{\max}}^h \end{bmatrix}.$$

and its solution may be expressed in terms of Laplace transforms as

$$\begin{bmatrix} \boldsymbol{X}^h(\alpha_h) \\ \boldsymbol{S}_2^h(\alpha_h) \\ \vdots \\ \boldsymbol{S}_{K_{\max}}^h(\alpha_h) \end{bmatrix} = (\alpha_h\boldsymbol{I} - \boldsymbol{V}^h)^{-1} \begin{bmatrix} \bar{\boldsymbol{x}}^h(0) \\ \bar{\boldsymbol{s}}_2^h(0) \\ \vdots \\ \bar{\boldsymbol{s}}_{K_{\max}}^h(0) \end{bmatrix}.$$

Recalling that due to the reset rule $\bar{\boldsymbol{s}}_2^h(0) = \ldots = \bar{\boldsymbol{s}}_{K_{\max}}^h(0) = \boldsymbol{0}$, the above expression implies that $\boldsymbol{X}^h(\alpha_h) = \boldsymbol{C}^h\bar{\boldsymbol{x}}^h(0)$ and $\boldsymbol{C}^h = [(\alpha_h\boldsymbol{I} - \boldsymbol{V}^h)^{-1}]_{M\times M}$, where $[\cdot]_{I\times J}$ is the sub-matrix defined by the first I rows and J columns of its argument. Using the last expression and again invoking the reset rule $\bar{\boldsymbol{x}}_c(0) = \bar{\boldsymbol{n}}_c(0)$, (39) may be rewritten as

$$\bar{\boldsymbol{n}}_{c+1}(0) = \boldsymbol{P}\,(\boldsymbol{I} + \boldsymbol{BC})\,\bar{\boldsymbol{n}}_c(0), \tag{40}$$

where $\boldsymbol{C} = diag(\boldsymbol{C}^1, \ldots, \boldsymbol{C}^E)$.

Lemma 1 *$\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})$ has the following block structure*

$$\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC}) = \begin{pmatrix} \boldsymbol{0} & p_{12}(\boldsymbol{I} + \boldsymbol{B}^2\boldsymbol{C}^2) & \ldots & p_{1E}(\boldsymbol{I} + \boldsymbol{B}^E\boldsymbol{C}^E) \\ p_{21}(\boldsymbol{I} + \boldsymbol{B}^1\boldsymbol{C}^1) & \boldsymbol{0} & \ldots & p_{2E}(\boldsymbol{I} + \boldsymbol{B}^E\boldsymbol{C}^E) \\ \vdots & \vdots & \ddots & \vdots \\ p_{E1}(\boldsymbol{I} + \boldsymbol{B}^1\boldsymbol{C}^1) & p_{E2}(\boldsymbol{I} + \boldsymbol{B}^2\boldsymbol{C}^2) & \ldots & \boldsymbol{0} \end{pmatrix},$$

where the matrices $(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h)$ are column stochastic.

Proof 3 *The sparsity and block structure follow immediately from the definition of $\boldsymbol{P}$, since $p_{hh} = 0$, for all stages h. To prove that $(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h)$ is column*

stochastic, let us observe that by the given definitions

$$\boldsymbol{B}^h = (-\boldsymbol{I} + (\boldsymbol{R}^h)^T \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1^h + \sum_{k=2}^{K_{max}} ((\boldsymbol{R}^h)^T \boldsymbol{\Phi}_k^h \boldsymbol{\mu}_k^h + \alpha_h \boldsymbol{I}) \prod_{j=k\ldots 2} (\alpha_h \boldsymbol{I} + \boldsymbol{\mu}_j^h)^{-1} (\boldsymbol{I} - \boldsymbol{\Phi}_{j-1}^h)\boldsymbol{\mu}_{j-1}^h$$

$$\boldsymbol{C}^h = \left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h - \boldsymbol{B}^h\right)^{-1}$$

where $\boldsymbol{D}^h = \sum_{k=2}^{K_{max}} \alpha_h \prod_{j=k\ldots 2} (\alpha_h \boldsymbol{I} + \boldsymbol{\mu}_j^h)^{-1} (\boldsymbol{I} - \boldsymbol{\Phi}_{j-1}^h)\boldsymbol{\mu}_{j-1}^h$ *is a diagonal matrix. Carrying out the products yields after some manipulations*

$$\begin{aligned}(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h) &= ((\boldsymbol{C}^h)^{-1} + \boldsymbol{B}^h)\boldsymbol{C}^h = \left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h\right)\boldsymbol{C}^h \\ &= \left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h\right)\left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h - \boldsymbol{B}^h\right)^{-1} = \boldsymbol{I} + \boldsymbol{B}^h\left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h - \boldsymbol{B}^h\right)^{-1}\end{aligned}$$

Let $\mathbf{1}^T$ *be a row-vector of ones, we get*

$$\mathbf{1}^T(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h) = \mathbf{1}^T + \mathbf{1}^T\boldsymbol{B}^h\left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h - \boldsymbol{B}^h\right)^{-1} = \mathbf{1}^T$$

since $(\boldsymbol{B}^h)^T$ *is an infinitesimal generator and thus* $\mathbf{1}^T\boldsymbol{B}^h = \mathbf{0}^T$.

We are thus left to prove that all entries of $(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h)$ *are non-negative. We consider again the expression* $(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h) = (\alpha_h \boldsymbol{I} + \boldsymbol{D}^h)\left(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h - \boldsymbol{B}^h\right)^{-1}$. *Now observe that* $\boldsymbol{D}^h$ *is a diagonal non-negative matrix, thus* $\alpha_h \boldsymbol{I} + \boldsymbol{D}^h - \boldsymbol{B}^h$ *is a M-matrix [34]. Since the inverse of a non-singular M-matrix is non-negative and* $(\alpha_h \boldsymbol{I} + \boldsymbol{D}^h)$ *is also non-negative, we conclude that all entries of* $(\boldsymbol{I} + \boldsymbol{B}^h\boldsymbol{C}^h)$ *are non-negative.* □

Theorem 2 *In light load, the blending algorithm has a unique fixed point equal to the dominant right eigenvector of* $\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})$.

Proof 4 *Using Lemma 1, we see that* $\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})$ *is a non-negative matrix, being the product of a row-stochastic matrix with a column-stochastic matrix. We now wish to determine the spectral radius of* $\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})$ *that characterises the convergence of* (40) *and then determine uniqueness using the Perron-Frobenius theorem. First observe that since* (40) *is an exact reformulation of the ODE system* (19)-(21)*, it preserves the fluid mass of the* $\overline{\boldsymbol{n}}_c(0)$ *vectors for all iterations* $c \geq 1$*, and hence their 1-norm. This implies that* $\rho(\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})) \geq 1$ *since a spectral radius less than unity would lead to* $\lim_{c \to +\infty} \overline{\boldsymbol{n}}_c(0) = \mathbf{0}$*, contrary to conservation of mass. Suppose now that* $\rho(\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})) > 1$*. Then* $||(\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC}))^k||_p$ *would not be bounded for increasing* k *under any p-norm. However this is not possible due to the structure of the matrices involved: this is because, by Lemma 1, powers of* $\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{BC})$ *are composed of blocks that are convex combinations, through the coefficients* $p_{h1}, \ldots, p_{hE}$*, of products of column-stochastic matrices. Since products of column-stochastic matrices are also column-stochastic matrices, we conclude*

that column entries in any power of $\boldsymbol{P}(\boldsymbol{I}+\boldsymbol{BC})$ *will be upper bounded by the numbers of blocks in each column, i.e., E. Thus,* $||(\boldsymbol{P}(\boldsymbol{I}+\boldsymbol{BC}))^k||_1 \leq E$, $\forall k$, *contrary to the assumption that* $\rho(\boldsymbol{P}\,(\boldsymbol{I}+\boldsymbol{BC})) > 1$. *This proves that* $\rho(\boldsymbol{P}\,(\boldsymbol{I}+\boldsymbol{BC})) = 1$.

Assuming that $\boldsymbol{P}$ *and the routing probability matrices* $\boldsymbol{R}$ *are irreducible, then* $\boldsymbol{P}\,(\boldsymbol{I}+\boldsymbol{BC})$ *is also irreducible and by the Perron-Frobenius theorem, its dominant eigenvalue is* 1 *(being equal to the spectral radius), it is unique, and its normalised right eigenvector has strictly positive entries.* □

Expression (40) can be recognised as the power method iteration used for stationary analysis of Markov chains, but applied to the matrix $\boldsymbol{P}\,(\boldsymbol{I}+\boldsymbol{BC})$. Note that the power method is guaranteed to converge, although its convergence rate can be slower than other eigenvalue algorithms [40].

7.2 Convergence analysis in heavy-load

In the heavy-load regime we assume that

$$\mathcal{L}(\min(\overline{\boldsymbol{n}}_c^h(t), \boldsymbol{S}^h); \alpha) = \mathcal{L}(\boldsymbol{S}^h; \alpha) = \alpha^{-1}\boldsymbol{S}^h.$$

This assumption models a heavy-loaded regime where all servers are busy at all stations. Using the heavy-load assumption in (32) and considering the transform of (18), it is found that

$$\overline{\boldsymbol{n}}_{c+1}(0) = \boldsymbol{P}\overline{\boldsymbol{n}}_c(0) + \boldsymbol{PBS}_{1,c}, \tag{41}$$

where $\boldsymbol{S}_{1,c} = (\boldsymbol{S}_{1,c}^1(\alpha_1), \ldots, \boldsymbol{S}_{1,c}^E(\alpha_E))$ is the Laplace transform vector for the $\bar{\boldsymbol{s}}_{1,c}^h(t)$ trajectories. We now make the observation that, under the reset rule, since all servers are continuously utilised, the trajectories $s_{1,c}(t)$ are independent of $\overline{\boldsymbol{n}}_c(0)$ and identical at each iteration. Thus

$$\overline{\boldsymbol{n}}_{c+1}(0) = \boldsymbol{P}\overline{\boldsymbol{n}}_c(0) + \boldsymbol{PBS}_{1,1}, \tag{42}$$

This is a first-order matrix difference equation with solution

$$\overline{\boldsymbol{n}}_c(0) = \boldsymbol{P}^c\overline{\boldsymbol{n}}_1(0) + \sum_{j=1}^{c} \boldsymbol{P}^j\boldsymbol{BS}_{1,1}, \tag{43}$$

Since $\boldsymbol{P}$ is a stochastic matrix, different cases are possible depending on $\boldsymbol{P}$ and the choice of the initial point.

As an example of a case where blending does not converge in heavy-load, consider a balanced network, for example a cyclic network of exponential stations

	$M=2, E=2, S=1$			$M=2, E=2, S=5$		
		Runtime [s]			*Runtime [s]*	
Dist.	*Error*	*SIM*	*BLE*	*Error*	*SIM*	*BLE*
Erl	0.062	806	75	0.062	366	47
Exp	0.078	99	6	0.020	27	6
Cox	0.075	398	11	0.024	204	10

Table 2
Validation results: networks with 2 stations and 2 stages.

having the same rates in any one stage, but which have different rates in different stages. We assume an identical number of servers S at each station, thus $\boldsymbol{S}_{1,1} = S\mathbf{1}$. In these models, it is possible to verify from the definitions that $\boldsymbol{B}$ is doubly-stochastic and thus $\boldsymbol{B}\boldsymbol{S}_{1,1} = S\boldsymbol{B}\mathbf{1} = \mathbf{0}$. This implies that the heavy-load iteration has a simplified structure

$$\overline{\boldsymbol{n}}_{c+1}(0) = \boldsymbol{P}\overline{\boldsymbol{n}}_{c}(0)$$

with initial vector $\overline{\boldsymbol{n}}_1(0)$. It is well-known from the Markov chain literature that an iteration of this kind may not converge, for example in cases where $\boldsymbol{P}$ is cyclic. We tested a cyclic $\boldsymbol{P}$ and $\overline{\boldsymbol{n}}_1^1(0) \neq \overline{\boldsymbol{n}}_1^2(0) \neq \ldots \neq \overline{\boldsymbol{n}}_1^E(0)$ confirming that the solution oscillates. Conversely, if the same model is initialised with the invariant eigenvector, i.e., $\overline{\boldsymbol{n}}_1^1(0) = \overline{\boldsymbol{n}}_1^2(0) = \ldots = \overline{\boldsymbol{n}}_1^E(0)$, the algorithm converges remaining at this fixed point. This suggests that in some instances the initial points may affect the convergence of the blending algorithm. Still, the procedure we have outlined in Section 6.3 for calculation of performance measures is robust, as it guarantees that blending returns an estimate also in instances where the iteration does not converge.

8 Numerical Validation

To study the error behaviour of the blending algorithm, we consider 6 groups of networks with randomly generated parameters. The model parameters are drawn from uniform distributions with rates in $[0.0, 50.0]$ for the random environment. Within a group, we keep fixed the number of queues (M), the number of stages in the random environment (E), and the number of servers in each queue (S). The population is equal to $N = 50$ jobs. Service time distributions are varied in a controlled form. For each such network, we consider 3 possible service distributions: exponential, Erlang-3 (squared coefficient of variation $c^2 = 1/3$), and two-stage Coxian distribution with balanced phase

	$M=2, E=10, S=1$			$M=2, E=10, S=5$		
		Runtime [s]			*Runtime [s]*	
Dist.	*Error*	*SIM*	*BLE*	*Error*	*SIM*	*BLE*
Erl	0.208	6188	15	0.127	5322	18
Exp	0.152	609	4	0.058	166	5
Cox	0.139	1150	19	0.039	1284	23

Table 3
Validation results: networks with 2 stations and 10 stages.

	$M=10, E=2, S=1$			$M=10, E=2, S=5$		
		Runtime [s]			*Runtime [s]*	
Dist.	*Error*	*SIM*	*BLE*	*Error*	*SIM*	*BLE*
Erl	0.105	64923	17359	0.024	29284	8673
Exp	0.116	5923	1555	0.025	1873	434
Cox	0.158	18238	4127	0.028	14504	1141

Table 4
Validation results: networks with 10 stations and 2 stages.

probabilities ($c^2 = 10$). The service distribution has mean chosen at random uniformly in $[0.001, 10.0]$.

For each group we have analysed 300 models, thus 1800 models were considered overall. Each model was analysed by simulation, stopped after the 95% confidence intervals were within 5% of the mean. Both simulation and the blending algorithm were implemented in MATLAB, in order to use the same environment to compare their execution times. The implementation of blending alternates execution of a non-stiff solver (`ode45`) and a stiff solver (`ode15s`) until both methods cannot further improve the estimate of the fixed point. We use an iteration tolerance $\delta_{max} = 0.01$. The maximum number of iterations is set to $C = 100$. To speed-up convergence, we use a decomposition approximation to determine the initial vectors $\overline{\boldsymbol{n}}_1^h(0)$. Such decomposition considers each stage h in isolation and returns the value of $\overline{\boldsymbol{n}}^h(t)$ for large t obtained by the fluid ODEs. This allows the initial point to accumulate mass on the bottleneck station for each stage, often saving some iterations to the blending algorithm. The time to generate these estimates is included in the figures shown in the tables.

The results are shown in Tables 2-4. For a model with a population of N jobs,

the accuracy error is expressed as in [12], which normalises the absolute error with respect to the total job population as follows:

$$Error = \max_{1 \leq i \leq N} \frac{|Q_i^{\mathrm{BLE}} - Q_i^{\mathrm{SIM}}|}{N},$$

Here Q_i^{BLE} and Q_i^{SIM} are the estimates of the mean queue length at station i computed by blending and by simulation, respectively. Each table shows the average errors and execution times for a set of 100 randomly generated networks, with controlled service time distributions and server multiplicities. The computational cost of the study is approximately equal to 1 month on two 16-core Xeon E5540 2.53Ghz, most of which spent on the simulations.

The following observations arise from the results:

- The mean accuracy of blending is adequate in all cases, with the error decreasing with bigger network topologies and more stages of the random environment.
- Blending is generally faster than simulation. In the networks with two stations, the difference in runtimes is two orders of magnitude on average, but it tends to decrease with larger network topologies. In the groups of networks with 10 stations (see Table 4), an analysis of the behaviour of blending reveals that increased time of the analysis is mainly caused by a larger runtime per single iteration; this is due to the fact that the system of ODEs increases linearly with the number of stations and that larger problems may lead to stiffness. The number of iterations, not reported in the table, are found to be quite similar in all groups.
- By comparing the errors across the same row in every table, we find that increasing the number of servers at each station, while keeping all the other parameters unchanged, leads to an appreciable increase in the quality of the approximation. This is important for practical purposes, since the modelling of modern multi-core servers often requires to consider queues with multiple servers.
- The service-time distributions have an impact on the accuracy error, but this is model-dependent and difficult to interpret. For example, the instances with Erlang-distributed service times incur the largest errors in Table 3, but they display the smallest errors in the groups of networks of Table 4. This may indicate that, overall, the choice of service distribution does not determine approximation accuracy as other parameters such as the number of servers.

Summarising, our study reveals that the blending algorithm is reasonably accurate in the majority of instances. Quite often, the largest errors are associated with stiff models or early termination ($c > C = 100$). The algorithm is faster than simulations run at 95% confidence intervals, a useful property for optimisation studies which may require the evaluation of thousands of models.

9 Related Work

With the exception of average-environment and decomposition, we are not aware of general analytical approximation techniques for closed queueing networks in random environments. The authors of [37] study the application of transient analysis for studying the behaviour of a queueing system in a Markov-modulated environment. Based on an analytical expression for the transient probability distribution of an $M/M/1$ queue, numerical approximations are derived for the $MMPP/M/1$ queue at equilibrium. Results are found to be accurate. Our approach is different in three ways. First, we focus on random environments independent of the queueing model state. Next, we provide a general method for a class of closed queueing networks. Finally, our method is technically different because the blending algorithm uses a fluid limit approximation.

Also of relevance, and a particular application in fact, is queueing under breakdown and repair [11, 31] as well as recent work on queues with fluctuating loads [23]. Such works feature random environments similar to the ones illustrated in Section 2 and show commonalities with our approach. For example, [23] uses Laplace transforms to study the state trajectory and [18, 19] develop semi-numerical methods to evaluate related Laplace transforms in fluid queues with semi-Markovian inputs. [24] applies fluid analysis to open queues and outlines a moment-matching methodology for tandem networks. Compared to the present work, these papers focus on open models, whereas the focus of this paper is on closed queueing networks.

Fluid limits have been extensively studied for both discrete-time and continuous-time Markov chains (e.g., [5] and [28], respectively). A wealth of applications to specific case studies can be found in the literature; for instance to MAC protocols [8, 38], TCP protocols [1, 2], peer-to-peer networks [36], caching algorithms [27] and load balancing strategies [21, 22].

Important requirements for the existence of a deterministic limit are conditions on the nature of the transition rates. In some specific cases of stochastic processes, such as the queueing networks considered in this paper, it is possible to define a general framework of convergence which holds for a large class of models; typically this is due to the fact that the transition rates give rise to a vector field for the ODE model which is Lipschitz continuous globally. This result has been exploited for other classes of stochastic models, for instance those that are induced by high-level modelling languages such as process algebra [25, 41].

In all these cases, all the transition rates enjoy properties that are analogous to the density-dependent form discussed in Section 5. In our model, instead,

the transitions due to the random environment do not scale with increasing population levels, but are fixed. Although approaches for multi-scale stochastic processes are available [4], they cannot be applied to our model because they require a clear separation between *fast* and *slow* processes. In general, however, our model of random environments is quite general and allows transitions between stages to be in the same time scale as the network's service rates.

A suitable mathematical framework for defining our model could be that of Piece-wise Deterministic Markov Processes (PDMPs) [15], where continuous flows defined as systems of ODEs are *reset* by stochastic jumps distributed according to exponential distributions. Therefore, one would naturally associate changes in the random environment with such jumps, and the queueing network dynamics within one stage as a continuous flow by the fluid model. It would be possible in principle to show that, with such an association, a suitable family of queueing networks with random environment converges to a PDMP [9]. The dynamics of a PDMP is characterised by a partial differential equation defining the joint probability distribution of the process being in one (discrete) stage and the continuous buffer being less than a certain level. Unlike our blending algorithm, such a model is very difficult to solve analytically in general, but can be simulated. An exception is [13], where an elegant solution is provided that exploits the decoupling between the stochastic and the deterministic dynamics.

10 Conclusion

Blending is a new technique for the analysis of closed queueing networks in Markovian random environments. The approach iteratively evaluates the mean number of jobs at each queue observed at the instants when the active stage changes. It does so by applying a fluid approximation to the queueing network model, conditioning its evolution on a given environment stage. This information is used to approximate the equilibrium behaviour of the system at steady state. We have used a large evaluation study to suggest strongly that the algorithm has good scalability and accuracy.

Several possible extensions of this work may be investigated, in particular multi-class closed networks and open models. While an extension to multi-class models appears possible, possibly under some more restrictive conditions on the scheduling disciplines, it is unclear at present if the blending algorithm could also be applied to open models. This is because the convergence analysis utilises a dependence on the property of conservation of the fluid mass, which exists because of the closed nature of the system.

Another direction for future work lies in extending the approach to systems

without the reset rule introduced in Section 3. For example, it would be valuable to investigate models where the number of phases in a service process is independent of the current stage or reset rules that instantaneously move jobs to other stations. Finally, it would be interesting to extend the validation to compare the blending algorithm against other approximation techniques.

Acknowledgement

The work of Giuliano Casale and Peter Harrison is supported by the European project MODAClouds (FP7-318484). The work of Mirco Tribastone is partially supported by the EU project QUANTICOL, 600708, and by the DFG project FEMPA. The authors thank Juan Fernando Peréz for helpful comments during the revision of this work.

References

[1] M. Alizadeh, A. Javanmard, and B. Prabhakar. Analysis of DCTCP: stability, convergence, and fairness. In *Proceedings of ACM SIGMETRICS 2011*, pages 73–84, 2011.

[2] M. Alizadeh, A. Kabbani, B. Atikoglu, and B. Prabhakar. Stability analysis of qcn: the averaging principle. In *Proceedings of ACM SIGMETRICS 2011*, pages 49–60, 2011.

[3] V. V. Anisimov. *Switching Processes in Queueing Models*. Wiley, 2008.

[4] K. Ball, T. Kurtz, L. Popovic, and G. Rempala. Asymptotic analysis of multiscale approximations to reaction networks. *Ann. Appl. Probab.*, 14(4):1925–1961, 2006.

[5] M. Benaïm and J.-Y. Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 65(11-12):823–838, 2008.

[6] P. Billingsley. *Probability and Measure*. Wiley, 3rd edition, 1995.

[7] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. 2nd ed., John Wiley and Sons, 2006.

[8] C. Bordenave, D. McDonald, and A. Proutiére. A particle system in interaction with a rapidly varying environment: Mean field limits and applications. *NHM*, 5(1):31–62, 2010.

[9] L. Bortolussi. Limit behavior of the hybrid approximation of stochastic process algebras. In *Proceedings of 17th International Conference on Analytical and*

Stochastic Modeling Techniques and Applications, ASMTA 2010, volume 6148 of *Lecture Notes in Computer Science*, pages 367–381. Springer, 2010.

[10] A. Chaintreau, J.-Y. L. Boudec, and N. Ristanovic. The age of gossip: spatial mean field regime. In *SIGMETRICS/Performance*, pages 109–120, 2009.

[11] R. Chakka and I. Mitrani. Heterogeneous multiprocessor systems with breakdowns: performance and optimal repair strategies. *Theoretical Computer Science*, 125(1):91–109, 14 Mar. 1994.

[12] K. M. Chandy and D. Neuse. Linearizer: A heuristic algorithm for queueing network models of computing systems. *Commun. ACM*, 25(2):126–134, 1982.

[13] F. Clévenot and P. Nain. A simple model for the analysis of squirrel. In *Proceedings of INFOCOM 2004*, 2004.

[14] P. Courtois. Decomposability, instabilities, and saturation in multiprogramming systems. *Comm. of the ACM*, 18(7):371–377, 1975.

[15] M. Davis. *Markov Models and Optimization.* Chapman & Hall, 1993.

[16] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. Ruge, and G. Sanders. Smoothed aggregation multigrid for Markov chains. *SIAM Journal on Scientific Computing*, 32(1):40–61, 2010.

[17] A. Economou. Generalized product-form stationary distributions for markov chains in random environments with queueing applications. *Advances in Applied Probability*, 37(1):pp. 185–211, 2005.

[18] A. J. Field and P. G. Harrison. An approximate compositional approach to the analysis of fluid queue networks. *Perform. Eval.*, 64(9-12):1137–1152, 2007.

[19] A. J. Field and P. G. Harrison. Busy periods in fluid queues with multiple emptying input states. *Journal of Applied Probability*, 47(2):474–497, 06 2010.

[20] J. M. Fourneau, B. Plateau, and W. Stewart. Product form for stochastic automata networks. In *Proc. of ValueTools*, pages 1–10, 2007.

[21] A. Ganesh, S. Lilienthal, D. Manjunath, A. Proutiere, and F. Simatos. Load balancing via random local search in closed and open systems. In *Proceedings of ACM SIGMETRICS 2010*, pages 287–298, 2010.

[22] N. Gast and B. Gaujal. A Mean Field Model of Work Stealing in Large-Scale Systems. *ACM SIGMETRICS*, 2010.

[23] V. Gupta, M. Harchol-Balter, A. Scheller-Wolf, and U. Yechiali. Fundamental characteristics of queues with fluctuating load. In *SIGMETRICS/Performance*, pages 203–215, 2006.

[24] V. Gupta and P. G. Harrison. Fluid level in a reservoir with an on-off source. *SIGMETRICS Performance Evaluation Review*, 36(2):128–130, 2008.

[25] R. Hayden and J. T. Bradley. A fluid analysis framework for a Markovian process algebra. *Theoretical Computer Science*, 2010.

[26] P. Heidelberger and P. D. Welch. A spectral method for confidence interval generation and run length control in simulations. *Comm. of the ACM*, 24(4):233–245, 1981.

[27] S. Ioannidis, L. Massoulié, and A. Chaintreau. Distributed caching over heterogeneous mobile networks. *Queueing Syst.*, 72(3-4):279–309, 2012.

[28] T. G. Kurtz. Solutions of ordinary differential equations as limits of pure Markov processes. *J. Appl. Prob.*, 7(1):49–58, April 1970.

[29] T. G. Kurtz. The relationship between stochastic and deterministic models for chemical reactions. *The Journal of Chemical Physics*, 57(7):2976–2978, 1972.

[30] I. Mitrani. *Probabilistic Modelling.* Cambridge University Press, 1998.

[31] I. Mitrani and A. A. Puhalskii. Limiting results for multiprocessor systems with breakdowns and repairs. *Queueing Syst*, 14(3-4):293–311, 1993.

[32] M. Mitzenmacher. *The power of two choices in randomized load balancing.* PhD thesis, University of California at Berkeley, Department of Computer Science, Berkeley, CA, 1996.

[33] M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models, an Algorithmic Approach.* The Johns Hopkins University Press, Baltimore, MD, 1981.

[34] R. J. Plemmons. M-matrix characterizations. I—nonsingular M-matrices. *Linear Algebra and Its Applications*, 18:175–188, 1977.

[35] P. Pollet. On a model for interference between searching insect parasites. *J. Austral. Math. Soc. Ser. B*, 32(2):133–150, 1990.

[36] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proceedings of ACM SIGCOMM 2004*, pages 367–378, 2004.

[37] P. Romano, B. Ciciani, A. Santoro, and F. Quaglia. Fast computation of hyper-exponential approximations of the response time distribution of MMPP/M/1 queues. In *Annual Simulation Symposium*, pages 113–120. IEEE Computer Society, 2008.

[38] G. Sharma, A. Ganesh, and P. Key. Performance analysis of contention based medium access control protocols. *IEEE Transactions on Information Theory*, 55(4):1665–1682, 2009.

[39] A. Stefanek, R. A. Hayden, and J. T. Bradley. GPA - A tool for fluid scalability analysis of massively parallel systems. In *QEST*, pages 147–148. IEEE Computer Society, 2011.

[40] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, NJ, USA, 1994.

[41] M. Tribastone, S. Gilmore, and J. Hillston. Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.*, 38(1):205–219, 2012.

[42] B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi. An analytical model for multi-tier internet services and its applications. In *Proc. of ACM SIGMETRICS*, pages 291–302. ACM Press, 2005.

[43] J. Zahorjan, E. D. Lazowska, and R. L. Garner. A decomposition approach to modelling high service time variability. *Performance Evaluation*, 3:35–54, 1983.

[44] Y. Zhu. Markovian queueing networks in a random environment. *Operations Research Letters*, (1):11 – 17.

A Markovian Random Environments

In this appendix, we illustrate under what conditions the average-environment and decomposition approximations are appropriate for modelling systems evolving in random environments. The results in this section are not limited to queueing networks and hold for any discrete-state Markov model.

Similarly to Section 2, we focus on an environment composed of two stages with infinitesimal generator

$$\boldsymbol{E} = \begin{bmatrix} -\alpha_{12} & \alpha_{12} \\ \alpha_{21} & -\alpha_{21} \end{bmatrix},$$

We consider a system evolving in it with infinitesimal generator $\boldsymbol{Q}_1^*$ in stage 1 and $\boldsymbol{Q}_2^*$ in stage 2. Since the environment is unaffected by the system state, the Markov process for the entire model has generator

$$\boldsymbol{Q} = \left[\begin{array}{c|c} \boldsymbol{Q}_1^* - \alpha_{12}\boldsymbol{I} & \alpha_{12}\boldsymbol{I} \\ \hline \alpha_{21}\boldsymbol{I} & \boldsymbol{Q}_2^* - \alpha_{21}\boldsymbol{I} \end{array}\right],,$$

where the diagonal blocks may have different orders. Let the equilibrium distribution of the network be the probability vector $\boldsymbol{\pi} = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2]$ such that $\boldsymbol{\pi}\boldsymbol{Q} = \boldsymbol{0}$, $\boldsymbol{\pi}\boldsymbol{1} = 1$. Here, $\boldsymbol{\pi}_1$ represents the equilibrium distribution vector for the states in $\boldsymbol{Q}_1^*$ corresponding to environment stage 1. Similarly, $\boldsymbol{\pi}_2$ is the equilibrium distribution for $\boldsymbol{Q}_2^*$ corresponding to stage 2. We have the following necessary condition.

Proposition 2 *At equilibrium*

$$\boldsymbol{\pi}_1 \left(\left(\frac{\alpha_{21}}{\alpha_{12}+\alpha_{21}} \right) \boldsymbol{Q}_1^* + \left(\frac{\alpha_{12}}{\alpha_{12}+\alpha_{21}} \right) \boldsymbol{Q}_2^* - \frac{\boldsymbol{Q}_1^*\boldsymbol{Q}_2^*}{\alpha_{12}+\alpha_{21}} \right) = \boldsymbol{0}, \tag{A.1}$$

$$\boldsymbol{\pi}_2 \left(\left(\frac{\alpha_{21}}{\alpha_{12}+\alpha_{21}} \right) \boldsymbol{Q}_1^* + \left(\frac{\alpha_{12}}{\alpha_{12}+\alpha_{21}} \right) \boldsymbol{Q}_2^* - \frac{\boldsymbol{Q}_2^*\boldsymbol{Q}_1^*}{\alpha_{12}+\alpha_{21}} \right) = \boldsymbol{0}. \tag{A.2}$$

Proof 5 *We first study the evolution of the Markov process* $\boldsymbol{Q}$ *embedded at instants where the stage changes from* 1 *to* 2. *Define* $\boldsymbol{\eta}_1^{(i)}$ *to be the exit probability distribution vector from* $\boldsymbol{Q}_1$ *after the* i*th transition from stage* 1 *to* 2. *Then for the Markov random environment under study*

$$\boldsymbol{\eta}_1^{(i)} = \boldsymbol{\eta}_1^{(i-1)} \int_0^{+\infty} \int_0^{+\infty} \alpha_{12} e^{(\boldsymbol{Q}_2 - \alpha_{21}\boldsymbol{I})b_i} \alpha_{21} e^{(\boldsymbol{Q}_1 - \alpha_{12}\boldsymbol{I})r_i} db_i dr_i, ,$$

for $i \geq 1$, *where* b_i *(resp.* r_i*) denotes the distribution of the* i*th elapsed time from entering stage 1 to the next breakdown (resp. stage 2 / repair) in the random environment. Using the fact that for a sub-generator* A

$$\int_0^{+\infty} e^{\boldsymbol{A}t} dt = (-\boldsymbol{A})^{-1}$$

we write

$$\boldsymbol{\eta}_1^{(i+1)} = \alpha_{12}\alpha_{21}\boldsymbol{\eta}_1^{(i)}(\boldsymbol{Q}_2 - \alpha_{21}\boldsymbol{I})^{-1}(\boldsymbol{Q}_1 - \alpha_{12}\boldsymbol{I})^{-1},$$

Observe now that a limiting vector $\boldsymbol{\eta}_1 = \lim_{i\to+\infty} \boldsymbol{\eta}_1^{(i)}$ *exists if* $\boldsymbol{Q}$ *admits an equilibrium distribution* $\boldsymbol{\pi}$, *since* $\boldsymbol{\eta}_1 = \boldsymbol{\pi}_1\alpha_{12}/\alpha_{12}^{tot}$, α_{12}^{tot} *being the aggregate arrival rate of repair events. (Note that for our particular random environment* $\alpha_{12}^{tot} = \alpha_{12}$*). Thus we have*

$$\boldsymbol{\eta}_1 = \alpha_{12}\alpha_{21}\boldsymbol{\eta}_1(\boldsymbol{Q}_2 - \alpha_{21}\boldsymbol{I})^{-1}(\boldsymbol{Q}_1 - \alpha_{12}\boldsymbol{I})^{-1}, ,$$

or equivalently since $\boldsymbol{\eta}_1 \propto \boldsymbol{\pi}_1$

$$\boldsymbol{\pi}_1 \left(\alpha_{12}\alpha_{21}\boldsymbol{I} - (\boldsymbol{Q}_1 - \alpha_{12}\boldsymbol{I})(\boldsymbol{Q}_2 - \alpha_{21}\boldsymbol{I})\right) = 0.,$$

Using a similar argument for the exit distribution $\boldsymbol{\eta}_2$ *after a breakdown, we may write*

$$\boldsymbol{\pi}_2 \left(\alpha_{12}\alpha_{21}\boldsymbol{I} - (\boldsymbol{Q}_2 - \alpha_{21}I)(\boldsymbol{Q}_1 - \alpha_{12}\boldsymbol{I})\right) = 0.,$$

The characterization (A.1)-(A.2) is then obtained straightforwardly after expanding the products and dividing both sides by $\alpha_{12} + \alpha_{21}$. □

This characterization highlights the relationship between the exact solution $\boldsymbol{\pi} = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2]$ and the average-environment and decomposition approximations. Note first that $\boldsymbol{Q}_1^*$ and $\boldsymbol{Q}_2^*$ are the generators used by the decomposition approximation for the model, which has solution $\boldsymbol{\pi}^{DEC} = [\boldsymbol{\pi}_1^{DEC}, \boldsymbol{\pi}_2^{DEC}]$ where

$$\begin{aligned} \boldsymbol{\pi}_1^{DEC}\boldsymbol{Q}_1^* &= \mathbf{0}, & \boldsymbol{\pi}_1^{DEC}\mathbf{1} = \pi_1 &= \frac{\alpha_{21}}{\alpha_{12}+\alpha_{21}}, \\ \boldsymbol{\pi}_2^{DEC}\boldsymbol{Q}_2^* &= \mathbf{0}, & \boldsymbol{\pi}_2^{DEC}\mathbf{1} = \pi_2 &= \frac{\alpha_{12}}{\alpha_{12}+\alpha_{21}}. \end{aligned}$$

Observing that the coefficients of $\boldsymbol{Q}_1^*$ and $\boldsymbol{Q}_2^*$ in (A.1)-(A.2) are the equilibrium

probabilities of the random environment for stages 1 and 2, we get

$$\boldsymbol{\pi}_1 \left(\boldsymbol{Q}^{AVG} - \frac{\boldsymbol{Q}_1^* \boldsymbol{Q}_2^*}{\nu} \right) = \mathbf{0} \tag{A.3}$$

$$\boldsymbol{\pi}_2 \left(\boldsymbol{Q}^{AVG} - \frac{\boldsymbol{Q}_2^* \boldsymbol{Q}_1^*}{\nu} \right) = \mathbf{0} \tag{A.4}$$

where $\nu = \alpha_{12} + \alpha_{21}$ and

$$\boldsymbol{Q}^{AVG} = \left(\frac{\alpha_{21}}{\alpha_{12} + \alpha_{21}} \right) \boldsymbol{Q}_1^* + \left(\frac{\alpha_{12}}{\alpha_{12} + \alpha_{21}} \right) \boldsymbol{Q}_2^*$$

is the AVG approximation generator for the model under study. Denote by $\boldsymbol{\pi}^{AVG}$, $\boldsymbol{\pi}_1^{AVG}\mathbf{1} = 1$, the equilibrium probability vector of $\boldsymbol{Q}^{AVG}$. Then, as the aggregate event rate ν decreases, the solution of the queueing network approaches that of the decomposition approximation, up to scaling factors, i.e.,

$$\boldsymbol{\pi}_1 \lim_{\nu \to 0^+} \left(\boldsymbol{Q}^{AVG} - \frac{1}{\nu} \boldsymbol{Q}_1^* \boldsymbol{Q}_2^* \right) = \mathbf{0} \Rightarrow \quad \boldsymbol{\pi}_1 \boldsymbol{Q}_1^* \boldsymbol{Q}_2^* = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\pi}_1 \propto \boldsymbol{\pi}_1^{DEC}$$

$$\boldsymbol{\pi}_2 \lim_{\nu \to 0^+} \left(\boldsymbol{Q}^{AVG} - \frac{1}{\nu} \boldsymbol{Q}_2^* \boldsymbol{Q}_1^* \right) = \mathbf{0} \Rightarrow \quad \boldsymbol{\pi}_2 \boldsymbol{Q}_2^* \boldsymbol{Q}_1^* = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\pi}_2 \propto \boldsymbol{\pi}_2^{DEC}$$

Conversely, if ν becomes asymptotically large one finds that the AVG solution becomes exact:

$$\boldsymbol{\pi}_1 \lim_{\nu \to 0^+} \left(\boldsymbol{Q}^{AVG} - \frac{1}{\nu} \boldsymbol{Q}_1^* \boldsymbol{Q}_2^* \right) = \mathbf{0} \Rightarrow \quad \boldsymbol{\pi}_1 \boldsymbol{Q}^{AVG} = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\pi}_1 = \boldsymbol{\pi}^{AVG}$$

$$\boldsymbol{\pi}_2 \lim_{\nu \to 0^+} \left(\boldsymbol{Q}^{AVG} - \frac{1}{\nu} \boldsymbol{Q}_2^* \boldsymbol{Q}_1^* \right) = \mathbf{0} \Rightarrow \quad \boldsymbol{\pi}_2 \boldsymbol{Q}^{AVG} = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\pi}_2 = \boldsymbol{\pi}^{AVG}$$

As expected, these results indicate that the average-environment and decomposition approximations provide exact results for models in random environments where the frequency of events is respectively very large or very small compared to the rates of the queues. However, in intermediate cases, we have established by (A.1)–(A.2) that the probability distributions $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ depend also on matrices $\boldsymbol{Q}_1^* \boldsymbol{Q}_2^*$ and $\boldsymbol{Q}_2^* \boldsymbol{Q}_1^*$, which are not infinitesimal generators, having negative off-diagonal entries. This suggests that there may *not* exist a simple way to approximate well in all cases $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ by weighting the equilibrium solutions for $\boldsymbol{Q}^{AVG}$, $\boldsymbol{Q}_1^*$, or $\boldsymbol{Q}_2^*$, a strategy that is common in previous work [14, 43].

Algorithm 1. Blending Algorithm

1: **input**:
- $\boldsymbol{R}^h$: routing matrix for all stages $h = 1, \ldots, E$
- $\boldsymbol{S}^h$: number servers for each of the M queues in stage h
- $\boldsymbol{x}_0^h$: initial queue-lengths estimates for the M queues in stage h
- $\boldsymbol{E}$: infinitesimal generator of the random environment
- $\boldsymbol{\mu}_k^h$, $k = 1, \ldots, K_{max}; h = 1, \ldots, E$
- $\boldsymbol{\Phi}_k^h$, $k = 1, \ldots, K_{max}; h = 1, \ldots, E$
- ϵ: numerical tolerance
- T: maximum integration time
- δ_{max}: convergence tolerance
- C: maximum number of iterations

2: **output**: approximate $\bar{\boldsymbol{n}}^h(t)$, $\bar{\boldsymbol{x}}^h(t)$, $\bar{\boldsymbol{s}}_k^h(t)$, $k = 1, \ldots, K_{max}$, $h = 1, \ldots, E$, $0 \le t \le T$

3: **algorithm**:

4: $c = 1$

5: **for** $h = 1, \ldots, E$ **do**

6: $\quad \bar{\boldsymbol{n}}_c^h(0) = \bar{\boldsymbol{x}}_c^h(0) = \boldsymbol{x}_0^h$,

7: $\quad \bar{\boldsymbol{s}}_{c,k}^h(0) = \boldsymbol{0}$, $k = 2, \ldots, K_{max}$

8: $\quad \boldsymbol{B}^h = (-\boldsymbol{I} + (\boldsymbol{R}^h)^T \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1^h + \sum_{k=2}^{K_{max}} ((\boldsymbol{R}^h)^T \boldsymbol{\Phi}_k^h \boldsymbol{\mu}_k^h + \alpha_h \boldsymbol{I}) \prod_{j=k\ldots 2} (\alpha_h \boldsymbol{I} + \boldsymbol{\mu}_j^h)^{-1} (\boldsymbol{I} - \boldsymbol{\Phi}_{j-1}^h)\boldsymbol{\mu}_{j-1}^h$

9: **end for**

10: compute $\boldsymbol{\pi}^{env} = [\pi_h^{env}]_{h=1,\ldots,E}$ such that $\boldsymbol{\pi}^{env}\boldsymbol{E} = \boldsymbol{\pi}^{env}$, $\sum_{h=1}^E \pi_h^{env} = 1$

11: compute $p_{he}^{src} = \pi_h^{env}\alpha_{he}\left(\sum_{j\neq e}\pi_j^{env}\alpha_{je}\right)^{-1}$, for $h, e = 1, \ldots, E; h \neq e$

12: **repeat**

13: $\quad c = c + 1$

14: $\quad$ **for** $h = 1, \ldots, E$ **do**

15: $\quad\quad$ solve for all $t \le T$, with numerical tolerance ϵ, the initial value problem

$$\frac{d\bar{\boldsymbol{x}}_c^h(t)}{dt} = (-\boldsymbol{I} + \boldsymbol{R}^T\boldsymbol{\Phi}_1)\boldsymbol{\mu}_1 \min(\bar{\boldsymbol{x}}_c^h(t), \bar{\boldsymbol{s}}_{c,1}^h(t)) + \textstyle\sum_{k=2}^{K_{max}} \boldsymbol{R}^T\boldsymbol{\Phi}_k\boldsymbol{\mu}_k\bar{\boldsymbol{s}}_{c,k}^h(t),$$
$$\frac{d\bar{\boldsymbol{s}}_{c,2}^h(t)}{dt} = -\boldsymbol{\mu}_2\bar{\boldsymbol{s}}_{c,2}^h(t) + (\boldsymbol{I} - \boldsymbol{\Phi}_1)\boldsymbol{\mu}_1 \min(\bar{\boldsymbol{x}}_c^h(t), \bar{\boldsymbol{s}}_{c,1}^h(t)),$$
$$\frac{d\bar{\boldsymbol{s}}_{c,k}^h(t)}{dt} = -\boldsymbol{\mu}_k\bar{\boldsymbol{s}}_{c,k}^h(t) + (\boldsymbol{I} - \boldsymbol{\Phi}_{k-1})\boldsymbol{\mu}_{k-1}\bar{\boldsymbol{s}}_{c,k-1}^h(t), \qquad k = 3, \ldots, K_{max}$$

$\quad\quad$ subject to

$$\bar{\boldsymbol{x}}_c^h(0) = \bar{\boldsymbol{n}}_c^h(0)$$
$$\bar{\boldsymbol{s}}_{c,1}^h(0) = \boldsymbol{S}^h,$$
$$\bar{\boldsymbol{s}}_{c,k}^h(0) = \boldsymbol{0}, \quad k = 2, \ldots, K_{max}$$
$$\bar{\boldsymbol{s}}_{c,1}^h(t) = \boldsymbol{S}^h - \textstyle\sum_{k=2}^{K_{max}} \bar{\boldsymbol{s}}_{c,k}^h(t), \quad t \ge 0$$

16: $\quad\quad$ compute $\boldsymbol{n}_c^h(t) = \bar{\boldsymbol{x}}_c^h(t) + \sum_{k=2}^{K_{max}} \bar{\boldsymbol{s}}_{k,c}^h(t)$, $0 \le t \le T$

17: $\quad\quad$ compute $\boldsymbol{M}_c^h = \int_0^T \bar{\boldsymbol{n}}_c^h(t)e^{-\alpha_h t}dt$

18: $\quad\quad$ compute $\bar{\boldsymbol{n}}_{c+1}^h(0) = \sum_{\substack{h=1\\h\neq h}}^E p_{he}^{src}\left(\bar{\boldsymbol{n}}_c^h(0) + \boldsymbol{B}^h\boldsymbol{M}_c^h\right)$,

19: $\quad$ **end for**

20: **until** $\max_{h=1..E} ||\bar{\boldsymbol{n}}_{c+1}^h(0) - \bar{\boldsymbol{n}}_c^h(0)||_1 < \delta_{\max}$ **or** $c \le C$

21: **return** $\bar{\boldsymbol{n}}_c^h(t)$, $\bar{\boldsymbol{x}}_c^h(t)$, $\bar{\boldsymbol{s}}_{c,k}^h(t)$, $k = 1, \ldots, K_{max}$, $0 \le t \le T$