Multi-server queueing systems with multiple priority classes

Mor Harchol-Balter* Takayuki Osogami† Alan Scheller-Wolf‡ Adam Wierman§

Abstract

We present the first near-exact analysis of an M/PH/k queue with $m > 2$ preemptive-resume priority classes. Our analysis introduces a new technique, which we refer to as Recursive Dimensionality Reduction (RDR). The key idea in RDR is that the m-dimensionally infinite Markov chain, representing the m class state space, is recursively reduced to a 1-dimensionally infinite Markov chain, that is easily and quickly solved. RDR involves no truncation and results in only small inaccuracy when compared with simulation, for a wide range of loads and variability in the job size distribution.

Our analytic methods are then used to derive insights on how multi-server systems with prioritization compare with their single server counterparts with respect to response time. Multi-server systems are also compared with single server systems with respect to the effect of different prioritization schemes – "smart" prioritization (giving priority to the smaller jobs) versus "stupid" prioritization (giving priority to the larger jobs). We also study the effect of approximating m class performance by collapsing the m classes into just two classes.

Keywords: M/GI/k, M/PH/k, multi-server queue, priority queue, matrix analytic methods, busy periods, multi-class queue, preemptive priority.

1 Introduction

Much of queueing theory is devoted to analyzing priority queues, where jobs (customers) are labeled and served in accordance with a priority scheme: high-priority jobs preempt medium-priority jobs, which in turn preempt low-priority jobs in the queue. Priority queueing comes up in many applications: Sometimes the priority of a job is determined by the job's owner via a Service Level Agreement (SLA), whereby certain customers have chosen to pay more so as to get high-priority access to some high-demand resource. Other

*Supported by NSF Career Grant CCR-0133077, NSF Theory CCR-0311383, NSF ITR CCR-0313148, and IBM Corporation via Pittsburgh Digital Greenhouse Grant 2003.
Email: harchol@cs.cmu.edu; Web: http://www.cs.cmu.edu/∼harchol; Address: Department of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, 15213; Phone: 412-268-7893; Fax: 412-268-5576.

†Email: osogami@cs.cmu.edu; Address: Department of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213; Phone: 412-268-3621.

‡Email: awolf@andrew.cmu.edu; Address: Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, 15213; Phone: 412-268-5066.

§Email: acw@cs.cmu.edu; Address: Department of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, 15213; Phone: 412-877-9455.

times, the priority of a job is artificially created, so as to maximize a company's profit or increase system utilization. For example, an online store may choose to give high-priority to the requests of big spenders, so that those customers are less likely to go elsewhere, see [18].

Analyzing the mean response time (and higher moments of response time) for different classes of jobs is clearly an important problem.[1] While this problem has been well understood in the case of a single-server M/GI/1 queue since the 1950's [5], the problem becomes much more difficult when considered in the context of a multi-server M/GI/k system, and even for an M/M/k system when jobs have different completion rates. This is unfortunate since such multi-server systems are prevalent in many applications where prioritization is used, e.g., web server farms and super-computing centers.

The reason that priority queueing is difficult to analyze in a multi-server setting is that jobs of different priorities may be in service (at different servers) at the same time, thus the Markov chain representation of the multi-class, multi-server queue appears to require tracking the number of jobs of each class. Hence one needs a Markov chain which is infinite in m dimensions, where m is the number of priority classes. While the analysis of a 1-dimensionally infinite Markov chain is easy, the analysis of an m-dimensionally infinite Markov chain ($m > 1$) is largely intractable.

Prior work

The number of papers analyzing multi-server priority queues is vast, however almost all are restricted to only *two priority classes*. Of those restricted to two priority classes, all assume *exponential service times*. The only papers *not* restricted to two priority classes are coarse approximations based on assuming that the multi-server behavior is related to that of a single server system [2] or approximations based on aggregating the many priority classes into two classes [20, 23].

Two priority classes

We start by describing the papers restricted to *two priority classes* and *exponentially distributed service demands*. Techniques for analyzing the M/M/k dual priority system can be organized into four types on which we elaborate below: (i) approximations via aggregation or truncation; (ii) matrix analytic methods; (iii) generating function methods; (iv) special cases where the priority classes have the same mean. Unless otherwise mentioned, preemptive-resume priorities should be assumed.

Nearly all analysis of dual priority M/M/k systems involves the use of Markov chains, which with two priority classes grows infinitely in two dimensions (one dimension for each priority class). In order to overcome this, researchers have simplified the chain in various ways. Kao and Narayanan truncate the chain by either limiting the number of high priority jobs [13], or the number of low priority jobs [11]. Nishida

[1] We will use the term *response time* throughout the paper to denote the time from when a job arrives until it is completed. We will also occasionally talk about the *delay* (wasted time), which we define as the job's response time minus its service requirement.

aggregates states, yielding an often rough approximation [23]. Kapadia, Kazmi and Mitchell explicitly model a finite queue system [14]. Unfortunately, aggregation or truncation is unable, in general, to capture the system performance as the traffic intensity grows large.

Although, in theory, the matrix analytic method can be used to directly analyze a 2D-infinite Markov chain (see for example [3]), the matrix analytic method is much simpler and more computationally efficient when it is applied to a 1D-infinite Markov chain. Therefore, most papers that use the matrix analytic method to analyze systems involving 2D-infinite Markov chains first reduce the 2D-infinite chain to a 1D-infinite chain by, for example, truncating the state space by placing an upper bound on the number of jobs [13, 11, 17, 22]. Miller [19] and Ngo and Lee [22] partition the state space into blocks and then "super-blocks," according to the number of high priority customers in queue. However, [19] experiences numerical instability issues when $\rho > 0.8$.

A third stream of research capitalizes on the exponential job sizes by explicitly writing out the balance equations and then finding roots via generating functions. In general these yield complicated mathematical expressions susceptible to numerical instabilities at higher loads. See King and Mitrani [20]; Gail, Hantler, and Taylor [8, 9]; Feng, Kowada, and Adachi [7]; and Kao and Wilson [12].

Finally there are papers that consider the special case where the multiple priority classes all have the same mean. These include Davis [6], Kella and Yechiali [15] (for non-preemptive priorities), and Buzen and Bondi [4].

The only work dealing with non-exponential service times is contained in a pair of papers, not yet published, by Sleptchenko et. al. [27, 28]. Both papers consider a two-priority, multi-server system where within each priority there may be a number of different classes, each with its own different exponential job size distribution. This is equivalent to assuming a *hyper-exponential job size distribution* for each of the *two priority classes*. The problem is solved via a combination of generating functions and matrix analytic methods. In theory, their technique may be generalizable to PH distributions, though they evaluate only hyper-exponential distributions due to the increased complexity necessary when using more general PH distributions.

More than two priority classes

For the case of *more than two priority classes*, there are only coarse approximations. The Bondi-Buzen (BB) approximation [2] is beautiful in its simplicity and usability. It is based on an intuitive observation that the "improvement" of priority scheduling over FCFS scheduling under k servers is similar to that for the case of one server with equal total capacity:

$$\frac{E[D^{\text{M/GI/}k\text{/prio}}]}{E[D^{\text{M/GI/}k\text{/FCFS}}]} \approx \frac{E[D^{\text{M/GI/1/prio}}]}{E[D^{\text{M/GI/1/FCFS}}]} = \text{ scaling factor.} \tag{1}$$

Here $E[D^{\text{M/GI/}k\text{/prio}}]$ is the overall mean delay under priority scheduling with k servers of speed $1/k$, and $E[D^{\text{M/GI/}k\text{/FCFS}}]$ is defined similarly for FCFS, while M/GI/1 refers to a single server queue with speed 1. This relation is exact when job sizes are exponential with the same rate for all classes; however what happens when this is not the case has never been established.

The other approximation (which we denote by MK-N) which allows for *more than two priority classes* and *exponential* job sizes is due to Mitrani and King [20], and also used by Nishida [23] to extend the latter author's analysis of two priority classes to $m > 2$ priority classes. The MK-N approximation analyzes the mean delay of the lowest priority class in an M/M/k queue with $m \geq 2$ priority classes by *aggregating all the higher priority classes*. Thus, instead of aggregating all jobs into one class, as BB does, MK-N aggregates into two classes. The job size distribution of the aggregated class is then approximated with an exponential distribution by matching the first moment of the distribution.

Contributions of this paper

In Section 2, we introduce a new analytical approach that provides the first near-exact analysis of the M/PH/k queue with $m \geq 2$ preemptive-resume priority classes. Our approach, which we refer to as Recursive Dimensionality Reduction (RDR), is very different from the prior approaches described above. RDR allows us to recursively reduce the m-dimensionally infinite state space, created by tracking the m priority classes, to a 1-dimensionally infinite state space, which is analytically tractable. The dimensionality reduction is done without any truncation; rather, we reduce dimensionality by introducing "busy period transitions" within our Markov chain, for various types of busy periods created by different job classes. The only approximation in the RDR method stems from the fact that we need to approximate these busy periods using Markovian (phase-type) PH distributions. We find that matching three moments of these busy periods is usually possible using a 2-phase Coxian distribution, and provides sufficient accuracy, within a couple percent of simulation, for all our experiments. Our experiments span load ranging from $\rho = 0.05$ to $\rho = 0.95$ and job size variability ranging from $C^2 = 1$ to $C^2 = 128$, where $C^2 = var(X)/(E[X])^2$, where X represents the job size (service requirement). The accuracy of the RDR method can be increased arbitrarily by better approximating the busy periods. Having reduced the problem to a 1-dimensionally infinite state space, we describe how to derive mean response time and higher moments of response time.

In theory RDR can handle systems with any number of servers, any number of priority classes, and PH service times. In addition, RDR is quite efficient; for all the scenarios explored in this paper, the computation time under RDR is less than a few seconds. However, the complexity of the RDR method does increase with both the number of servers k and the number of classes m. Because RDR becomes less practical under high m and k, we develop a much simpler, but only slightly less accurate, approximation RDR-A (see Section 2.5). RDR-A simplifies calculations by approximating an m priority system with a two priority system, which is then solved using RDR.

In Section 3 we present results from both RDR and RDR-A for per-class mean response time for an

M/PH/k queue with multiple priority classes. In Section 4, we use these results to obtain many interesting *insights* about priority queueing. First, in Section 4.1 we compare the performance of priority queueing in a multi-server system with k servers each of speed $1/k$ versus a single server of speed 1. We find that the effect of priorities in a single server system can be very different than in a multi-server system of equal capacity. (A non-surprising consequence of this result is that the BB approximation, which relies on relating a multi-server system to a single server system, can exhibit large errors.) Next, in Section 4.2, we study the effect of priority policies that favor short jobs ("smart prioritization") versus priority policies that favor long jobs ("stupid prioritization") under systems with different numbers of servers. Understanding the effect of "smart" prioritization is important because many common scheduling policies are designed to give priority to short jobs. Lastly, in Section 4.3, we ask how effective class aggregation (aggregating $m > 2$ priority classes into just 2 priority classes) is as an approximation for dealing with systems having more than two priority classes. We evaluate two types of class aggregation, that used in the MK-N approximation and that used in RDR-A, to show when class aggregation serves as a reasonable approximation.

2 RDR analysis of M/PH/k with m priority classes

In this section we describe the RDR technique, dividing our explanation into three parts. As an introduction, in Section 2.1, we deal only with the simplest case of $m = 2$ priority classes and exponential job sizes, which we solve using the techniques in [26, 30]. We then move to the difficult case of $m > 2$ priority classes, but still with exponential service times, in Section 2.2. Here the techniques from [26, 30] do not apply, so we introduce Recursive Dimensionality Reduction (RDR). The RDR approach uses the analysis of the $m - 1$ priority classes to analyze the m-th priority class. This is a non-trivial procedure for $m > 2$ since it involves evaluating many complex passage times (busy periods) in the chain representing the $m - 1$ priority classes, as these passage times now form transitions within the chain representing m priority classes. Finally in Section 2.3, we show how RDR can be applied to the most general case of PH service times with $m > 2$ priority classes.

All the analysis up to through Section 2.3 deals with how to derive mean per-class response times. In Section 2.4 we illustrate how the RDR method can be extended to obtain *variance of response time* for each class. Finally, in Section 2.5, we introduce RDR-A, which is an approximation of RDR, allowing very fast (< 1 second) evaluation of high numbers of priority classes and servers, with small ($< 5\%$) error.

2.1 Simplest case: Two priority classes, exponential job sizes

Consider the simplest case of two servers and two priority classes, high (H) and low (L), with exponentially distributed sizes with rates μ_H and μ_L respectively. Figure 1(a) illustrates a Markov chain of this system, whose states track the number of high priority and low priority jobs; hence this chain grows infinitely in two dimensions. Observe that high priority jobs simply see an M/M/2 queue, and thus their mean response

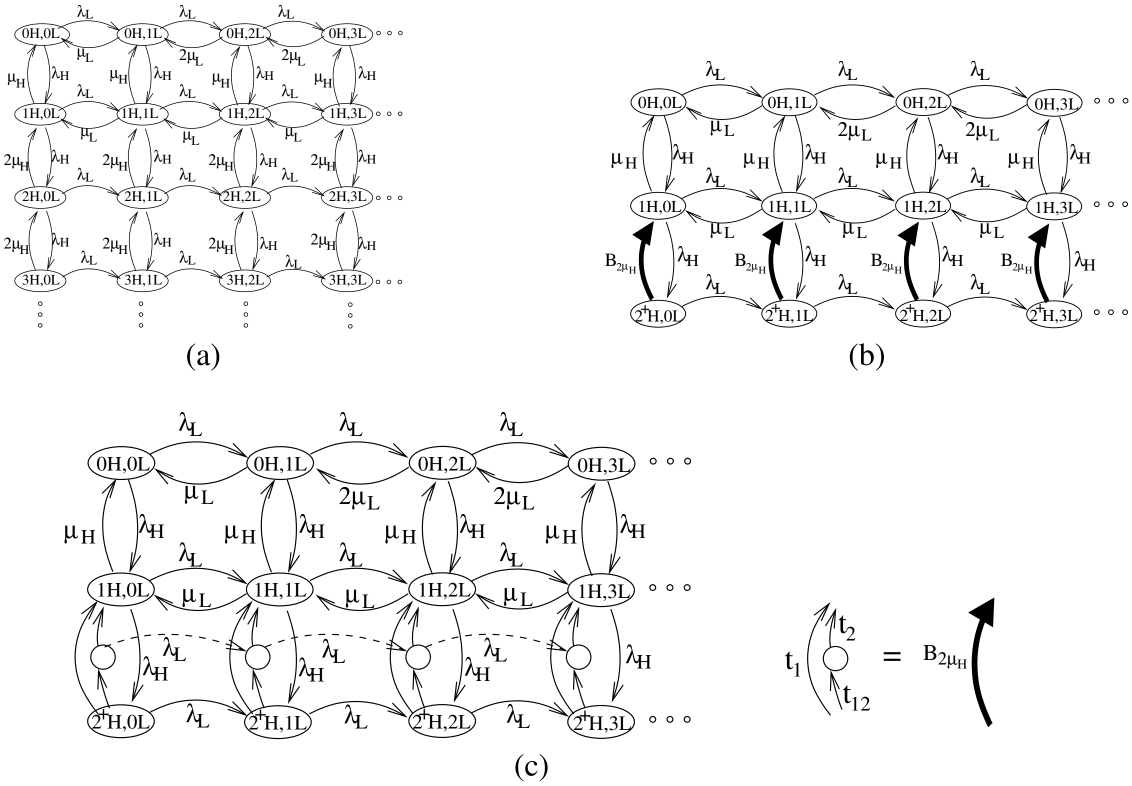


Figure 1: *(a) Markov chain for an M/M/2 queue with two priority classes. This Markov chain is infinite in two dimensions. Via the Dimensionality Reduction technique, we arrive at the chain in (b), which uses busy period transitions, and is only infinite in one dimension. In (b), the busy period is represented by a single transition. In (c), the busy period is represented by a two phase PH distribution (with Coxian representation), yielding a 1D-infinite Markov chain.*

time is well-known. Low priority jobs, however, have access to either an M/M/2, M/M/1, or no server at all, depending on the number of high priority jobs. Thus their mean response time is more complicated, and this is where we focus our efforts.

Figure 1(b) illustrates the reduction of the 2D-infinite Markov chain to a 1D-infinite chain. The 1D-infinite chain tracks the number of low priority jobs exactly. For the high priority jobs, the 1D-infinite chain only differentiates between zero, one, and two-or-more high priority jobs. As soon as there are two-or-more high priority jobs, a *high priority busy period* is started. During the high priority busy period, the system only services high priority jobs, until the number of high priority jobs drops to one.[2] The length of time spent in this high priority busy period is exactly an M/M/1 busy period where the service rate is $2\mu_H$. We denote the duration of this busy period by the transition labeled $B_{2\mu_H}$.

[2]Throughout the paper a "higher priority busy period" is defined as the time from when the system has k higher priority jobs until there are only $k-1$ higher priority jobs.

The busy period $B_{2\mu_H}$ is *not* exponentially-distributed. Hence it is not clear how it should fit into a Markov model. We use a PH distribution (specifically a Coxian distribution) to match the first three moments of the distribution of $B_{2\mu_H}$. Parameters of the PH distribution, whose first three moments match those of $B_{2\mu_H}$, are calculated via the closed form solutions provided in [25].

Figure 1(c) illustrates the same 1D-infinite chain as in Figure 1(b), except that the busy period transition is now replaced by a two phase PH distribution with parameters t_1, t_{12} and t_2. The limiting probabilities in this 1D-infinite chain can be analyzed using matrix analytic methods [16]. These in turn yield the mean number of low-priority jobs, which via Little's law yields the mean response time for low-priority jobs. The only inaccuracy in the above approach is that only three moments of the high-priority busy period are matched. We will see later that this suffices to obtain very high accuracy across a wide range of load and job size distributions.

More formally, the 1D-infinite Markov chain is modeled as a (nonhomogeneous) QBD process, where level ℓ of the process denotes the ℓ-th column, namely all states of the form (iH,ℓL) for each ℓ. The generator matrix, $\mathbf{Q}$, of this process can be expressed as a block diagonal matrix:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F}^{(0)} & & & \\ \mathbf{B}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & & \\ & \mathbf{B}^{(2)} & \mathbf{L}^{(2)} & \mathbf{F}^{(2)} & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

where submatrix $\mathbf{F}^{(\ell)}$ encodes (forward) transitions from level (column) ℓ to level $\ell+1$ for $\ell \geq 0$, submatrix $\mathbf{B}^{(\ell)}$ encodes (backward) transitions from level ℓ to level $\ell-1$ for $\ell \geq 1$, and submatrix $\mathbf{L}^{(\ell)}$ encodes (local) transitions within level ℓ for $\ell \geq 0$. Specifically, for the Markov Chain depicted in Figure 1(c), we order the 4 states in level ℓ as: $(0H, \ell L)$, $(1H, \ell L)$, $(2^+H, \ell L)$, $(xH, \ell L)$. The pair of states $\{(2^+H, \ell L), (xH, \ell L)\}$ are the states used to represent the busy period, where the state $(2^+H, \ell L)$ denotes the start of the busy period and $(xH, \ell L)$ denotes the intermediate "bubble" state in the busy period. Given this ordering of states, we have:

$$\mathbf{L}^{(\ell)} = \left(\begin{array}{cc|cc} -\sigma_1 & \lambda_H & & \\ \mu_H & -\sigma_2 & \lambda_H & \\ \hline & t_1 & -\sigma_3 & t_{12} \\ & t_2 & & -\sigma_4 \end{array}\right)$$

where $\sigma_i = \sum_{j \neq i} \mathbf{Q}_{ij}$. The delineations of the matrix $\mathbf{L}^{(\ell)}$ are intended to highlight the busy period.

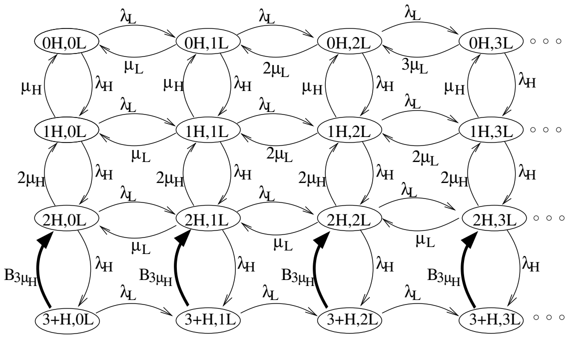


Figure 2: *This chain illustrates the case of two priority classes and three servers. The busy period transitions are replaced by a Coxian phase-type distribution matching three moments of the busy period duration, as shown in Figure 1.*

$\mathbf{F}^{(\ell)} = \lambda_L \mathbf{I}$, for $\ell \geq 0$, where $\mathbf{I}$ is a 4×4 identity matrix, and

$$\mathbf{B}^{(\ell)} = \mu_L \begin{pmatrix} \min(2,\ell) & & & \\ & 1 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}$$

for $\ell \geq 1$.

The stationary probabilities of being in level ℓ, $\vec{\pi}_\ell$, are then given recursively by $\vec{\pi}_\ell = \vec{\pi}_{\ell-1} \cdot \mathbf{R}^{(\ell)}$., where $\vec{\pi}_0$ and $\mathbf{R}^{(\ell)}$ are calculated as follows: Define $\hat{\ell}$ to be the level that the QBD process starts repeating (in Figure 1(c), $\hat{\ell} = 1$), then for $\ell = 1, \ldots, \hat{\ell}$, we have that $\mathbf{R}^{(\ell)}$ is given recursively by:

$$\mathbf{F}^{(\ell-1)} + \mathbf{R}^{(\ell)} \cdot \mathbf{L}^{(\ell)} + \mathbf{R}^{(\ell)} \cdot \mathbf{R}^{(\ell+1)} \cdot \mathbf{B}^{(\ell+1)} = \mathbf{0},$$

where $\mathbf{0}$ is a zero matrix of appropriate dimension (4×4). For $\ell \geq \hat{\ell} + 1$, $\mathbf{R}^{(\ell)} = \mathbf{R}$, where $\mathbf{R}$ is given by the minimal solution to the following matrix quadratic equation:

$$\mathbf{F}^{(\hat{\ell})} + \mathbf{R} \cdot \mathbf{L}^{(\hat{\ell})} + \mathbf{R}^2 \cdot \mathbf{B}^{(\hat{\ell})} = \mathbf{0}.$$

Vector $\vec{\pi}_0$ is given by a positive solution of $\vec{\pi}_0 \left(\mathbf{L}^{(0)} + \mathbf{R}^{(1)} \cdot \mathbf{B}^{(1)} \right) = \vec{0}$, normalized by the equation $\vec{\pi}_0 \sum_{\ell=0}^{\infty} \prod_{i=1}^{\ell} \mathbf{R}^{(i)} \cdot \vec{1} = 1$, where $\vec{0}$ and $\vec{1}$ are column vectors with an appropriate number of elements of 0 and 1, respectively. The mean response time can now be computed using the stationary distributions, $\vec{\pi}_\ell$'s, given above, via Little's law.

Figure 2 shows the generalization to a three server system. We simply add one row to the chain shown in Figure 1, and now differentiate between 0, 1, 2, or 3-or-more high priority jobs. This can be easily extended to the case of $k > 3$ servers.

2.2 Harder case: m priority classes, exponential job sizes

We now turn to the more difficult case of $m > 2$ priority classes. We illustrate this for the case of two servers and three priority classes: high-priority (H), medium-priority (M), and low-priority (L). The mean response time for class H jobs and that for class M jobs are easy to compute. Class H jobs simply see an M/M/2 queue. Class M jobs see the same system that the low-priority jobs see in an M/M/2 queue having two priority classes. Replacing the L's by M's in the chain in Figure 1 yields the mean response time for the M class jobs.

The analysis of the class L jobs is the difficult part. The obvious approach would be to aggregate the H and M jobs into a single class, so that we have a 2-class system (H-M versus L jobs). Then we could apply the technique of the previous section, tracking exactly the number of low-priority jobs and maintaining limited state information on the H-M class. This is the approach that we follow in Section 2.5 in deriving our RDR-A approximation. However, this approach is imprecise because the duration of the busy periods in the H-M class depends on whether the busy period was started by 2H jobs, 1H and 1M job, or 2M jobs in service. By ignoring the priorities among H's and M's, we are ignoring the fact that some types of busy periods are more likely than others. Even given the information on who starts the busy period, this still does not suffice to determine its duration, because the duration is also affected by the prioritization within the aggregated H-M class.

Thus, a precise response time analysis of class L requires maintaining more information. As before, we want to exactly track the number of class L jobs. Given that there are two servers, we need to further differentiate between whether there are zero H and M jobs, one H or M job, or two or more H and M jobs. Whenever there are two or more H and M jobs, we are in an H-M busy period. For an M/M/2 with three priority classes, there are *six types of busy periods* possible, depending on the state at the start of the busy period – $(2H, 0M)$, $(1H, 1M)$, or $(0H, 2M)$ – and the state in which the busy period ends – $(1H, 0M)$ or $(0H, 1M)$. We derive the busy period duration by conditioning on who starts and ends the busy period.

Figure 3 (left) shows the level of the 1D-infinite chain in which the number of class L jobs is u. In state (wH,vM,uL), v class M jobs and w class H jobs are in the system if $v + w < 2$; otherwise, the state (wH,vM,uL) denotes that we are in a H-M busy period that was started by v class M jobs and w class H jobs. Observe that there are six types of busy periods depicted, labeled B_1, B_2, $\ldots$, B_6; the busy period is determined by the state in which it was started and the state in which it ends. Notice, for example, that both states in the fourth and fifth row are labeled (0H,2M,uL), meaning that the busy period is started by two class M jobs; but these two states differ in the class of the job that is left at the end of the H-M busy period. In state (0H,2M,uL) of the fourth row, the busy period ends leaving a class H job, whereas in state of the fifth row, the busy period ends leaving a class M job. (Recall that the class of job left at the end of a busy period is probabilistically determined at the *beginning* of the busy period and the duration of the busy period is conditioned on the class of the job left at the end.) Here $p_{2M,H}$, for example, denotes the probability

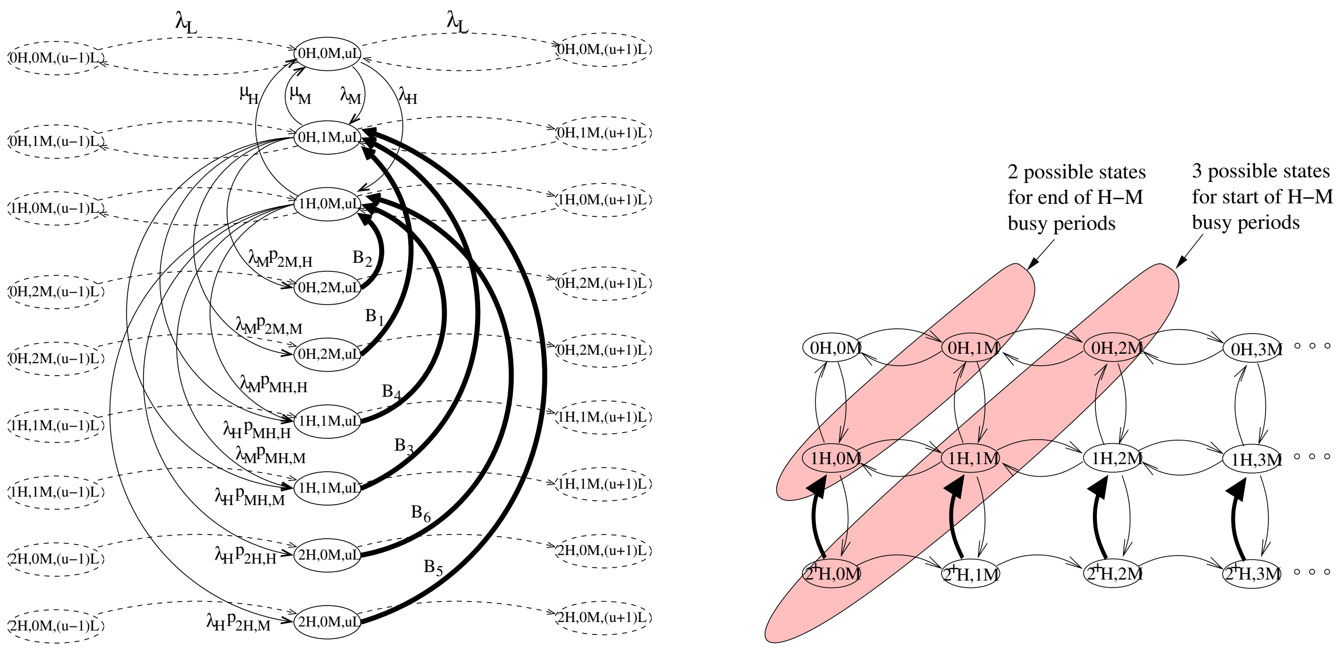


Figure 3: *(Left) Portion of the 1D-infinite chain used to compute mean response time for low-priority jobs in the case of three priority classes and two servers, and all exponential service times. (Right) Chain used to compute moments of the durations of the six busy period transitions.*

that the busy period started by two class M jobs ends leaving one class H job, whereas $p_{HM,M}$ denotes the probability that the busy period started by one class H and one class M job, ends leaving one class M job. The remaining probabilities are defined similarly.

It remains to derive the moments of the duration of busy periods, B_1, B_2, ..., B_6, and probabilities $p_{2M,M}$, $p_{2M,H}$, $p_{HM,M}$, $p_{HM,H}$, $p_{2H,M}$, and $p_{2H,H}$ in Figure 3(left). The trick to deducing these quantities is to observe that the six busy periods correspond to passage times between two "diagonal" (shaded) levels in the chain shown in Figure 3(right), which is the 1D-infinite chain that we used to analyze the class M performance. We refer to the right shaded diagonal level as level ℓ and the left shaded diagonal level as level $\ell - 1$ (where $\ell = 3$). Note that the 3 states in level ℓ correspond to the three possible "start" states for busy periods, and the two states in level $\ell - 1$ correspond to the two possible "end" states for the busy periods. Thus, for example, busy period B_1 in Figure 3(left) corresponds to the first passage time from state $(0H, 2M)$ to state $(0H, 1M)$ in the chain in Figure 3(right), given that $(0H, 1M)$ is the first state reached in level $\ell - 1$, when starting in state $(0H, 2M)$ of level ℓ. Likewise, probability $p_{2M,M}$ corresponds to the probability that, in Figure 3(right), state (0H,1M) is the first state of the two possible "end" states that is reached in level $\ell - 1$, given that the "start" state is $(0H, 2M)$. These conditional inter-level passage times and ending probabilities within the chain in Figure 3(right) can be calculated using techniques developed by Neuts in [21]. We provide a precise description of this in Appendix A. Observe that these computations are greatly facilitated by the fact that our chains are infinite in only one dimension.

More formally, the 1D-infinite Markov chain shown in Figure 3(left) is modeled as a (nonhomogeneous) QBD process, as in Section 2.1. Here, level ℓ of the QBD process denotes the ℓ-th column, namely all states of the form (iH,jM,ℓL) for each ℓ. The submatrices of the QBD process, $\mathbf{L}^{(\ell)}$, $\mathbf{F}^{(\ell)}$, and $\mathbf{B}^{(\ell)}$, have size 15×15. Specifically, the forward transitions are $\mathbf{F}^{(\ell)} = \lambda_L \mathbf{I}$, for $\ell \geq 0$, where $\mathbf{I}$ is a 15×15 identity matrix. The backward transitions are

$$\mathbf{B}^{(\ell)} = \mu_L \begin{pmatrix} \min(2,\ell) & & \\ & 1 & \\ & & \mathbf{0} \end{pmatrix}$$

for $\ell \geq 1$, where $\mathbf{0}$ is a zero matrix of size 13×13. We again order the 15 states in level ℓ as: $(0H, 0M, \ell L)$, $(0H, 1M, \ell L)$, $(1H, 0M, \ell L)$, followed by the two states associated by busy period B_i, for $i = 1, 2, \ldots, 6$, where, as before, the two states associated with each busy period are ordered by start state, intermediate state. The local transitions are then given by:

$$\mathbf{L}^{(\ell)} = \left(\begin{array}{ccc|c|c|c|c|c|c}
-\sigma_1 & \lambda_M & \lambda_H & & & & & & \\
\mu_M & -\sigma_2 & & \lambda_M \vec{p}^{(2M,M)} & \lambda_M \vec{p}^{(2M,H)} & \lambda_M \vec{p}^{(MH,M)} & \lambda_M \vec{p}^{(MH,H)} & & \\
\mu_H & & -\sigma_3 & & & \lambda_H \vec{p}^{(MH,M)} & \lambda_H \vec{p}^{(MH,H)} & \lambda_H \vec{p}^{(2H,M)} & \lambda_H \vec{p}^{(2H,H)} \\
\hline
 & \vec{t}^{(1)} & & \mathbf{T}^{(1)} & & & & & \\
\hline
 & & \vec{t}^{(2)} & & \mathbf{T}^{(2)} & & & & \\
\hline
 & \vec{t}^{(3)} & & & & \mathbf{T}^{(3)} & & & \\
\hline
 & & \vec{t}^{(4)} & & & & \mathbf{T}^{(4)} & & \\
\hline
 & \vec{t}^{(5)} & & & & & & \mathbf{T}^{(5)} & \\
\hline
 & & \vec{t}^{(6)} & & & & & & \mathbf{T}^{(6)}
\end{array}\right)$$

for $\ell \geq 0$, where the lines delineate the six busy periods ordered as $B_1, B_2, \ldots, B_6$, and where

$$\vec{t}^{(i)} = \begin{pmatrix} t_1^{(i)} \\ t_2^{(i)} \end{pmatrix}, \quad \mathbf{T}^{(i)} = \begin{pmatrix} -\sigma_{2i+2} & t_{12}^{(i)} \\ 0 & -\sigma_{2i+3} \end{pmatrix}, \quad \text{and } \sigma_i = \sum_{j\neq i} Q_{ij}$$

$$\vec{p}^{(X,Y)} = (p_{X,Y}, 0) \quad \text{where } X \in \{2H, HM, 2M\}, Y \in \{H, M\},$$

for $1 \leq i \leq 6$, corresponding to busy periods $B_1, B_2, \ldots, B_6$. Here $t_1^{(i)}, t_2^{(i)}, t_{12}^{(i)}$ are the rates of the PH distribution used to represent busy period B_i.

Now, the stationary probability of this QBD process can be calculated via matrix analytic methods, as in Section 2.1. The mean response time in the priority system can then be computed using the stationary distributions via Little's law.

The extension of RDR to $m > 3$ classes is straightforward but increases in complexity. For example, for the case of $m = 4$ classes, we proceed as in Figure 3, where we first create a chain that tracks exactly the number of jobs in class 4, and creates busy periods for the aggregation of the three higher priority classes. Then, to derive the busy periods for the three higher priority classes, we make use of the existing chain for three classes shown in Figure 3(left), and compute the appropriate passage times for that chain. For an M/M/k with m priority classes, there are $\binom{m+k-2}{k}\binom{m+k-3}{k-1}$ possible busy periods, where the first term in the product represents the number of possible start states (all combinations of up to $m-1$ priority classes over k servers) and the second term represents the number of possible end states (all combinations of up to $m-1$ priority classes over $k-1$ servers). That is, the number of different types of busy periods is polynomial in k if m is constant ($\Theta(k^m)$), and it is polynomial in m if k is constant ($\Theta(m^k)$); however, it is exponential in k and m if neither k nor m is constant.[3] Help with implementing the procedure described in this paper is provided at [24].

Practically speaking, the RDR approach is fast for a small number of servers and a small number of priority classes. In examples we ran with an M/M/2 and 10 priority classes, the RDR algorithm yielded mean response times within tens of seconds.

2.3 General case: Analysis of M/PH/k with m priority classes

In this section, we explicitly describe how RDR can be applied to analyze the case of PH job size distributions. We describe RDR for the case of two servers ($k = 2$) and two priority classes ($m = 2$), high (H) and low (L) , where the class H jobs have a particular 2-phase PH job size distribution with Coxian representation, shown in Figure 4(a).[4] Generalization to higher k's and higher m's is straightforward by applying the recursive algorithm introduced in Section 2.2.

Analyzing the performance of class H is trivial, since high-priority jobs simply see the mean response time in an M/PH/2 queue, which can be analyzed via standard matrix analytic methods. To analyze the class L jobs, as before, we create a 1D-infinite Markov chain tracking the class L jobs, and use busy period transitions to represent needed information regarding the class H jobs.

Observe that under the 2-phase Coxian job sizes distribution, we will need *four* different types of busy periods for high priority jobs, depending on the phases of the two jobs starting the busy period (1 & 1, or 1 & 2) and the phase of the job left at the end of the busy period (1 or 2). To derive the durations of these busy periods, we observe that the busy periods correspond to passage times from shaded level 3 to shaded level 2 in the Markov chain shown in Figure 4(b). Figure 4(b) describes the behavior of class H jobs, where states track the number of high priority jobs in the system and the phases of the jobs being processed. Namely,

[3] Remark: We note that in practice the number of busy periods can be reduced further, so that an M/M/k with m priority classes has $\binom{m+k-3}{k-1}^2$ busy periods of class 1 to class $m-1$ jobs. An advantage of this reduction is that the number of busy periods of class 1 to class $m-1$ jobs becomes independent of the type of PH distributions that is used to approximate the busy period of class 1 to class $m-2$ jobs. The trick to reducing the number of busy periods is illustrated by considering the example of the M/M/2 with three classes, shown in Figure 3. Here, by taking the mixture of the six busy periods, B_1, B_2, ..., B_6, we can approximate the H-M busy period by *four* PH distributions. These four distributions of the H-M busy period are differentiated by the state from which we *enter* the H-M busy period (either (1H,0M) or (0H,1M)) and by the state we return to after the H-M busy period (either (1H,0M) or (0H,1M)). More details are provided in [24].

[4] Under the Coxian job size distribution, a job starts in phase one where it is processed for a time exponentially distributed with rate $\mu_H^{(1)}$, and then either completes (with probability $q_H = 1 - p_H$) or moves to phase two (with probability p_H).

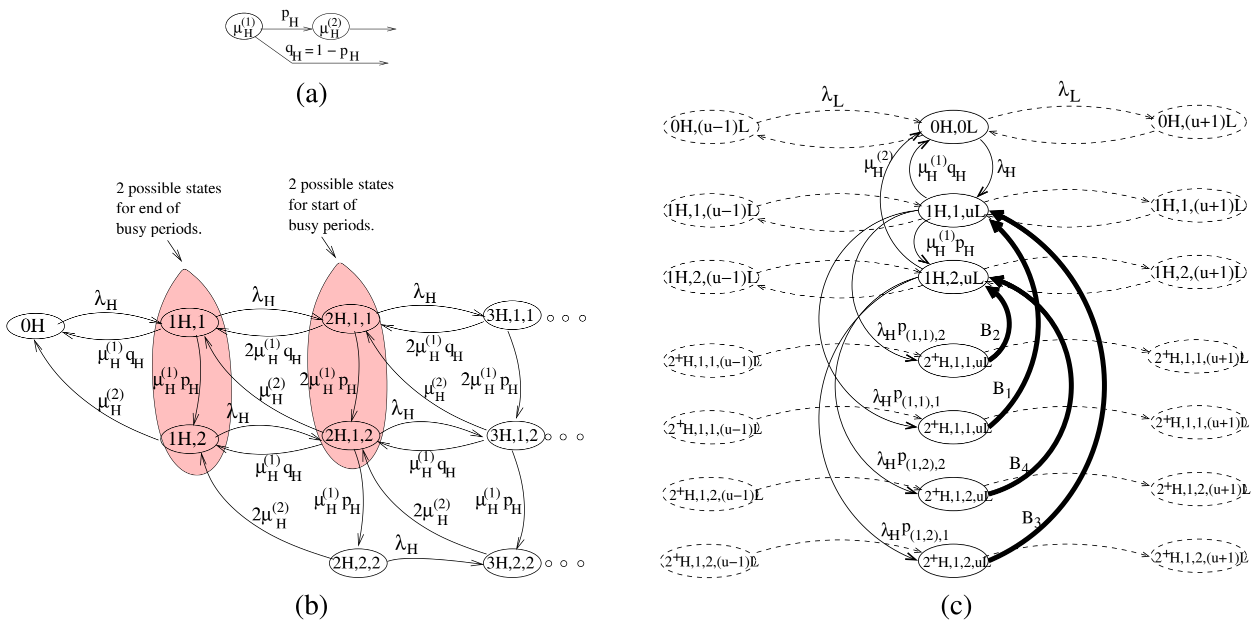


Figure 4: *(a) A 2-phase PH distribution with Coxian representation. (b) Markov chain which will be used to compute the high-priority job busy periods, in the case where high-priority job size have a PH distribution with Coxian representation shown in (a). (c) Chain for a system with two servers and two priority classes where high priority jobs have Coxian service times.*

at state (0H) there are no high priority jobs in the system; at state (1H,i), there is one high priority job in phase i; at state (nH,i, j) there are n high priority jobs in the system and the two jobs are being processed are in phase i and j, respectively (jobs in the queue are all in phase 1). The first passage times in Figure 4 are computed again using techniques in Appendix A.

Figure 4(c) shows a level of the chain that tracks the number of low priority jobs, where the number of low priority jobs is u. The low priority job sizes are assumed to be exponentially distributed, but this can be generalized to PH distributions. In state (0H,uL), no high priority jobs are in the system. An arrival of a high priority job in state (0H,uL) triggers a transition to state (1H,1,uL). In state (1H,j,uL), one high priority job in phase j is in service for $j = 1, 2$. An arrival of a high priority job in state (1H,j,uL) triggers a transition to state (2^+H,$1, j$,uL) for $j = 1, 2$. In state (2^+H,$1, j$,uL), at least two high priority jobs are in the system, and the two jobs that started the busy period were in phase 1 and j, respectively, for $j = 1, 2$. The four types of busy periods are labeled as B_1, B_2, B_3, and B_4, and the duration of these busy periods is approximated by PH distributions by matching the first three moments of the busy period distribution (note that the busy period cannot start with two jobs in phase two). Finally, $p_{(1,j),i}$ denotes the probability that a busy period started by two jobs in phases 1 and j, ends with a single job in phase i, for $j = 1, 2$, and $i = 1, 2$.

2.4 Computing variance of response time and higher moments

Throughout our discussion of RDR thus far, we have been concerned with computing the mean per-class response time. It turns out that computing higher moments of per-class response time is not much more difficult. Before we present our approach, we make two remarks. First, observe that it is trivial to derive all moments of the steady-state per-class *number of jobs* in the system, directly from the steady-state probabilities for the Markov chain, which we have already computed. Unfortunately, however, we cannot apply the beautiful generalization of Little's Law to higher moments (see [29, 1]) to immediately get the per-class higher moments of response time for free, since jobs do not necessarily leave our system in the order in which they arrive.

Below we sketch our approach for computing per-class variance in response time for the case of two servers, two priority classes (H and L), and exponential service times. We will refer to Figure 1(c) during our discussion. For class H jobs, it is easy to compute the variance of their response time, since they are oblivious to class L jobs, and the variance of response time in an M/M/2/FCFS queue is well known (see page 529 in [10]). Thus we will concentrate on class L jobs.

Consider the 1D-infinite Markov Chain shown in Figure 1(c) that tracks the number of class L jobs. We use the limiting probabilities to condition on what a class L arrival sees. Specifically, by PASTA (Poisson Arrivals See Time Averages) a class L arrival with probability $\pi_{(iH,\ell L)}$ will see state $(iH, \ell L)$ when it arrives, and will cause the system state to change to $(iH, (\ell + 1)L)$ at that moment.

To calculate the variance in response time seen by this "tagged" (class L) arrival, we remove all the λ_L arcs from the Markov chain in Figure 1(c), so that there are no more class L arrivals. This enables us to view the response time for the tagged arrival as the first passage time of this modified chain from state $(iH, (\ell + 1)L)$ to the state where the tagged arrival departs. The only complexity is in figuring out exactly in which state the tagged arrival departs.

The tagged arrival may depart the modified Markov chain the first time it hits $(0H, 1L)$ or $(1H, 1L)$, depending on the sample path the chain follows. We will compute the passage time to go from state $(iH, (\ell+1)L)$ to one of these states $\{$ $(0H, 1L)$ or $(1H, 1L)$ $\}$. It is important to observe that the first time we hit a state with 1L, cannot be state $(2^+H, 1L)$, by virtue of the fact that the Markov chain does not have decreasing arcs in its bottom rows.

If $(1H, 1L)$ is the first state that we hit with 1L, then we know that we must have gotten there from $(1H, 2L)$, which means that the single L job remaining is in fact the tagged arrival. (We're assuming preemption is done "youngest first to be preempted"). Thus we need to now add on the passage time to go from $(1H, 1L)$ to $(*, 0L)$ to get the full response time for the tagged arrival.

If $(0H, 1L)$ is the first state that we hit with 1L, then we know that we got there from state $(0H, 2L)$. In this case, the remaining 1L is equally likely to be the tagged arrival or not. With probability half, the tagged arrival is already gone, in which case we add nothing to the response time. With probability half, the tagged arrival remains, in which case we now add on the passage time to go from $(0H, 1L)$ to $(*, 0L)$ to get the full response time for the tagged arrival.

Observe that computing the moments of the above passage times is straightforward, since all the λ_L arcs have been removed.

2.5 Introducing RDR-A

We have seen that the RDR method can become computationally intensive as the number of priority classes grows. This motivates us to introduce an approximation based on RDR called RDR-A. RDR-A applies to $m > 2$ priority classes and PH job size distributions.

The key idea behind RDR-A is that the RDR computation is far simpler when there are only two priority classes: H and L. In RDR-A, under m priority classes, we simply aggregate these classes into two priority classes, where the $m-1$ higher priority classes become the new aggregate H class and the m^{th} priority class becomes the L class. We define the H class to have a PH job size distribution that matches the first three moments of the aggregation of the $m-1$ higher priority classes.

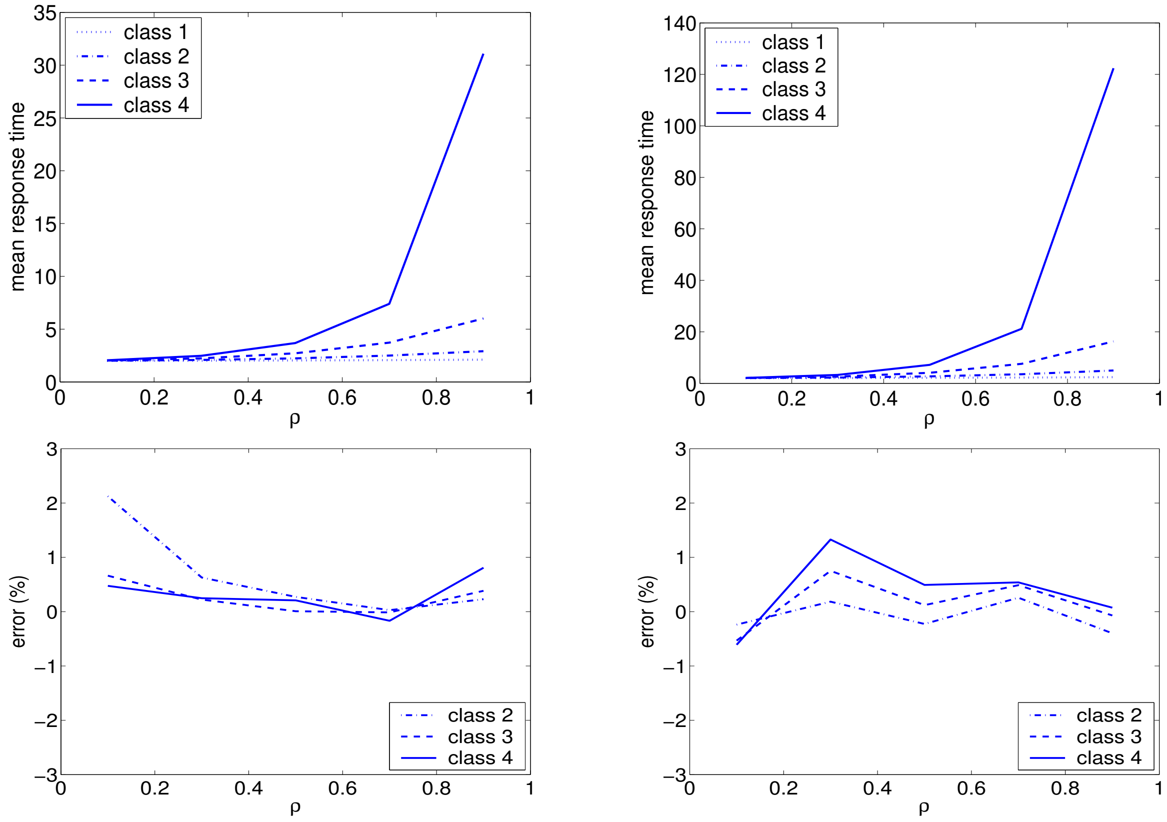
The RDR-A method is similar to the MK-N approximation. The difference is that in MK-N, both the H and L classes are exponentially distributed. Thus under MK-N, the H class only matches the *first* moment of the aggregate $m-1$ classes, whereas under RDR-A *three* moments are matched. The reason that we are able to match the first three moments, rather than just the first, is that we have the RDR technique, which allows the analysis of multi-server priority queues with *PH* job size distributions, as described in Section 2.3.

3 Results and Validation

In this section we present numerical results for per-class mean response times in M/M/k and M/PH/k queues with $m = 4$ priority classes, derived using RDR and RDR-A, respectively. We will validate our results against simulation and show that their relative error is small. Furthermore, the time required to generate our results is short, typically less than a second for each data point.

Figure 5 (top row) shows our results for per-class mean response times in an M/M/2 queue (left plot) and an M/PH/2 queue (right plot), both as a function of load ρ. The PH distribution used is a 2-phase PH distribution with squared coefficient of variation, $C^2 = 8$. All job classes have the same distribution, and the load is distributed evenly between the four classes. The left plot is derived using RDR and the right plot using RDR-A. Observe that the M/PH/2 queue (right plot) results in higher mean response time than the M/M/2 queue (left plot), as expected. In both cases the mean response time of the lower-priority classes dwarfs that of the higher-priority classes.

Figure 5 (bottom row) shows the relative per-class error for our results, when compared with simulation. Throughout the paper we always show error in *delay* (queueing time) rather than response time (sojourn



(a) M/M/2 with four classes

(b) M/PH/2 with four classes

Figure 5: *Top row shows per-class mean response time for M/M/2 (left) and M/PH/2 (right) with four priority classes. Left graph is derived using RDR and right graph is derived using RDR-A. Bottom row shows the error in our analytically-derived mean delay relative to simulation results, for the corresponding graphs in the top row.*

time), since the error in delay is proportionally greater. We define relative error as

$$\text{error} = 100 \times \frac{(\text{mean delay by RDR or RDR-A}) - (\text{mean delay by simulation})}{(\text{mean delay by simulation})} \quad (\%).$$

We only show the error for classes 2, 3, and 4, since our analysis is virtually exact for class 1 (solved via matrix-analytic methods). We see that the relative error in the mean delay of RDR and RDR-A compared to simulation is within 2% for all classes and typically within 1%, for all ρ's (the jaggedness of the figure is due to the fact that error is only evaluated at discrete loads). This error increases only slightly when we move to the case of priority classes with different means.

Figure 6 (left) again uses RDR-A to calculate per-class mean response time in the M/PH/2 queue with four classes, but this time as a function of C^2, the squared coefficient of variation of the job size distribution. (Again, all classes have the same job size distribution). As we see from the figure, the per-class mean response time increases nearly linearly with C^2. Figure 6 (right) shows the relative error in mean delay

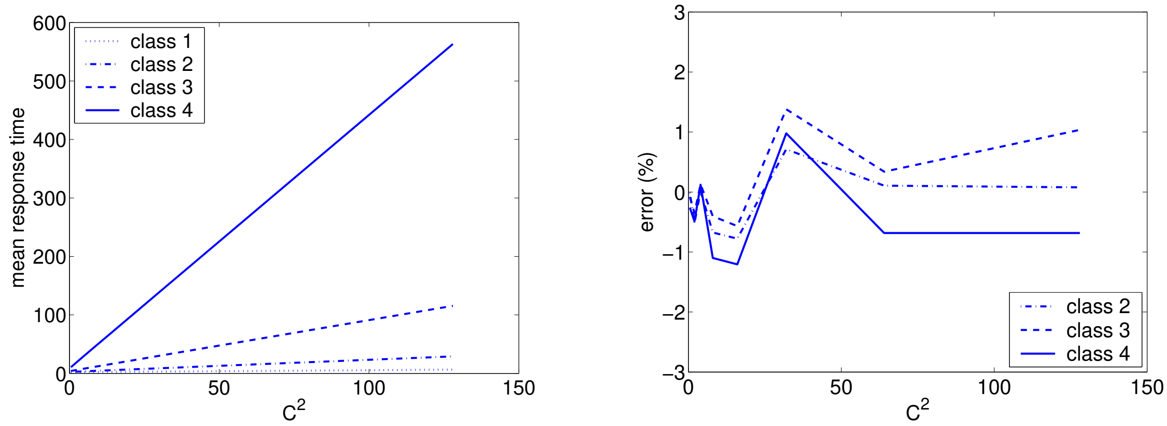


Figure 6: *(Left) Per-class mean response times for M/PH/2 with four priority classes, derived via RDR-A analysis. (Right) Relative error in analysis of mean delay compared with simulation.*

when the results of the RDR-A analysis in the left plot are compared with simulation. Again the error is under 2%. Again, this error increases only slightly when we move to the case of priority classes with different means.

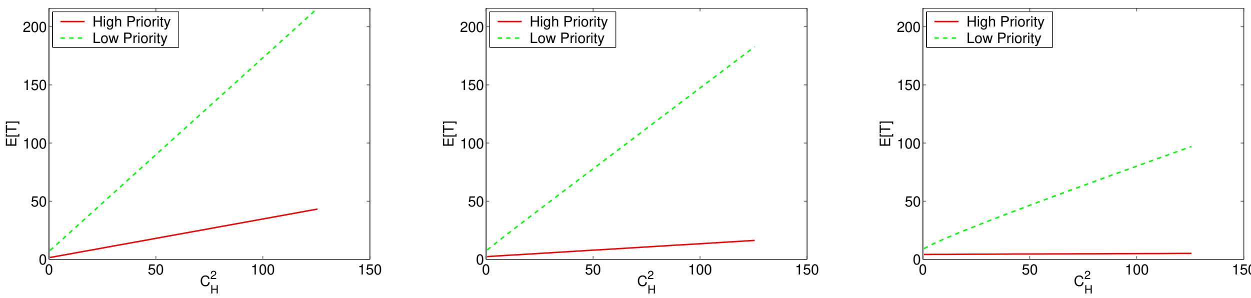
Finally, we note that in the above computations RDR is much more computationally efficient than simulation. Simulation requires tens of minutes to generate each figure, since the simulation is run 30 times, and in each run 1,000,000 events are generated. By comparison our analysis takes only a few seconds for each figure. Further, if we try to reduce the number of events in the simulation to 100,000 events, in order to speed it up, we see five times as much variation in the simulation around our analytical values.

4 Comparisons and Insights

In this section, we apply RDR and RDR-A to answer fundamental questions on prioritization in multi-server systems. In Section 4.1 we study the behavior of multi-server versus single server systems under prioritization. In this context, we also evaluate the BB approximation, which approximates the effect of prioritization in a multi-server system by that in a single server system. In Section 4.2, we evaluate the effect of prioritization schemes which favor short jobs in multi-server systems. Finally, in Section 4.3 we study the effect of aggregating multiple priority classes into just two classes, so as to significantly speed up the analysis. In this context we also evaluate the MK-N approximation.

4.1 Comparing multi-server versus single server performance under prioritization

In this section, we compare systems with different numbers of servers. It is important to note that throughout these comparisons, *we hold the total system capacity fixed.* That is, we compare a single server of unit speed with a 2-server system, where each server has speed half, with a 4-server system, where each server has



(1 server) (2 servers) (4 servers)

Figure 7: *Contrasting per-class mean response time under one server (left), two servers (middle) and four servers (right) for an M/PH/k with two priority classes. Total system capacity is fixed throughout, and* $\rho = 0.8$*. Results are obtained using RDR.*

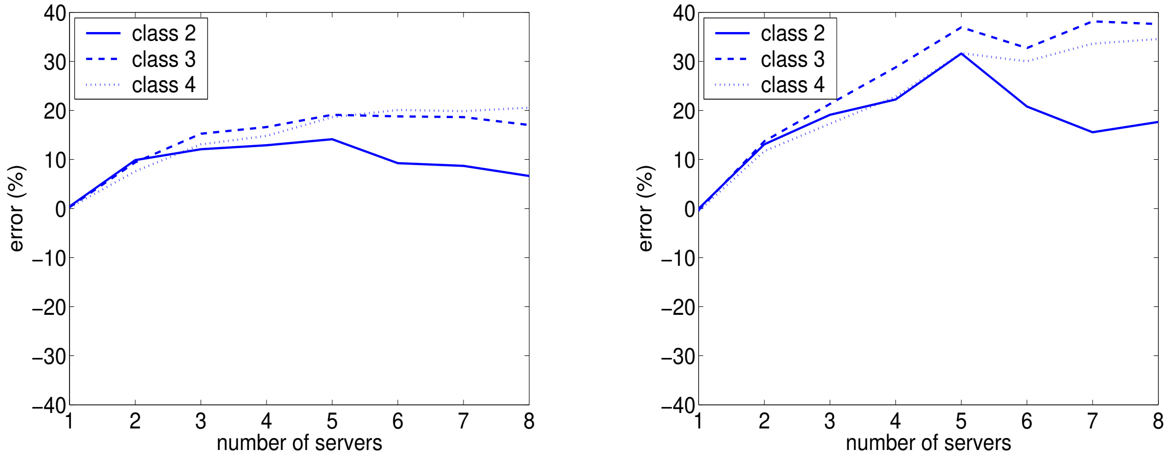
speed one-fourth, etc.

Figure 7 considers an M/PH/k system with two priority classes where k is one (left), two (middle), four (right), but the total system capacity is held fixed, and load is fixed at $\rho = 0.8$. The low-priority jobs are exponentially distributed. The high-priority jobs follow a PH distribution where the squared coefficient of variation for high priority jobs, C^2_H, is varied. The means of the two classes are the same and the load is split evenly between the two classes. The plots show per-class mean response time as a function of C^2_H. All results are computed using RDR.

The first thing to observe is that the response times in the case of one server appear very different from the response times in the case of two servers, or four servers. The effect of prioritization in a single server system offers little (quantitative) insight into the effect of prioritization in a multi-server system, aside from the fact that in all cases the response times appear to be a nearly linear function of C^2_H.

Figure 7 also illustrates some other interesting points. We see that as we increase the number of servers, under *high* C^2_H, the performance of both high-priority and low-priority jobs improves. By contrast, under *low* C^2_H, the performance can get worse as we increase the number of servers (this fact is more visible in Figure 7 for the high-priority jobs). To understand this phenomenon, observe that when C^2_H is high, short jobs can get stuck behind long jobs, and increasing the number of servers can allow the short jobs a chance to be served. By contrast when C^2_H is low, all jobs are similar in size, so we do not get the benefit of allowing short jobs to jump ahead of long jobs when there are more servers. However we do get the negative effect of increasing the number of servers, namely the underutilization of system resources when there are few jobs in the system, since each of the k servers only has speed $1/k$. The behavior under low C^2_H, where more servers lead to worse performance, is more prominent under lower load ρ.

Figure 7 already implies that the effect of prioritization on mean response time in a multi-server system may be quite different from that in a single server system. In Figure 8 we investigate this phenomena more closely, by evaluating the accuracy of the BB approximation [2], which is based on this assumption of similar behavior in single and multi-server priority queues. Looking at Figure 8, we see that the error in the



(a) $C^2 = 8$ (b) $C^2 = 25$

Figure 8: *Error in predicting mean delay using the BB approximation (compared with simulation) for an M/PH/2 with four classes where* $C^2 = 8$ *(left) or* $C^2 = 25$ *(right) and* $\rho = 0.8$.

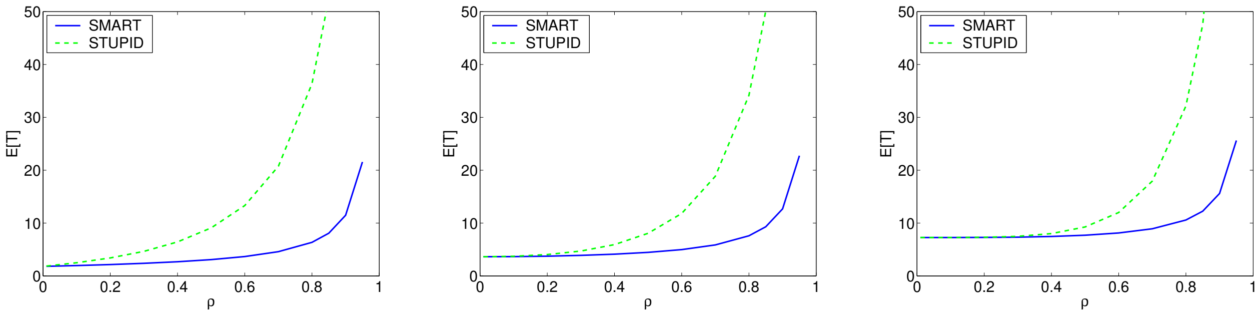
BB approximation appears to increase for higher C^2 (right graph) and for more classes. With four classes and two servers, the error is already 10% when $C^2 = 8$ and higher for higher C^2. By contrast, for the same 4-class case as shown in Figure 8, the error in RDR is always $< 2\%$ independent of C^2 and the number of servers (we have omitted this graph). In the above graphs all classes were statistically identical. In the case where the classes have different means, the error in BB can be much higher, whereas RDR-A is much less insensitive to this.

4.2 The effect of biasing toward short jobs in multi-server versus single server systems

Until now, we have assumed that all job classes have the same means. In this section and the next section, we remove this assumption. In this section we consider the effect of priority schemes which favor short jobs in multi-server systems. Biasing towards short jobs is a common method for improving mean response time. We use RDR to understand how the benefit of favoring short jobs in a single server system compares to that for a multi-server system.

Figure 9 considers a job size distribution comprised of an exponential of mean 1, representing jobs which are "short" in expectation, and an exponential of mean 10, representing jobs which are "long" in expectation (where job sizes are measured in a single-server system). The probability of each type of job is chosen to split load evenly between the short and long jobs (e.g., with $m = 2$ classes, $\frac{10}{11}$ of the jobs are short and $\frac{1}{11}$ of the jobs are long). The SMART scheduling policy assigns high priority to the short jobs, and the STUPID scheduling policy assigns high priority to the long jobs (possibly due to economic reasons). Figure 9 shows the results for a (a) one server, (b) two server, and (c) four server system.

Looking at Figure 9, the SMART and STUPID policies are the same when load ρ is low. At low load, the response time for both policies converges to simply the mean job size, which in these figures is $\frac{20}{11}$ for



(1 server) (2 servers) (4 servers)

Figure 9: *Mean response time under SMART versus STUPID prioritization in a 2-class system, where the classes are exponentially distributed with means one and ten respectively, for the case of one server, two servers, and four servers.*

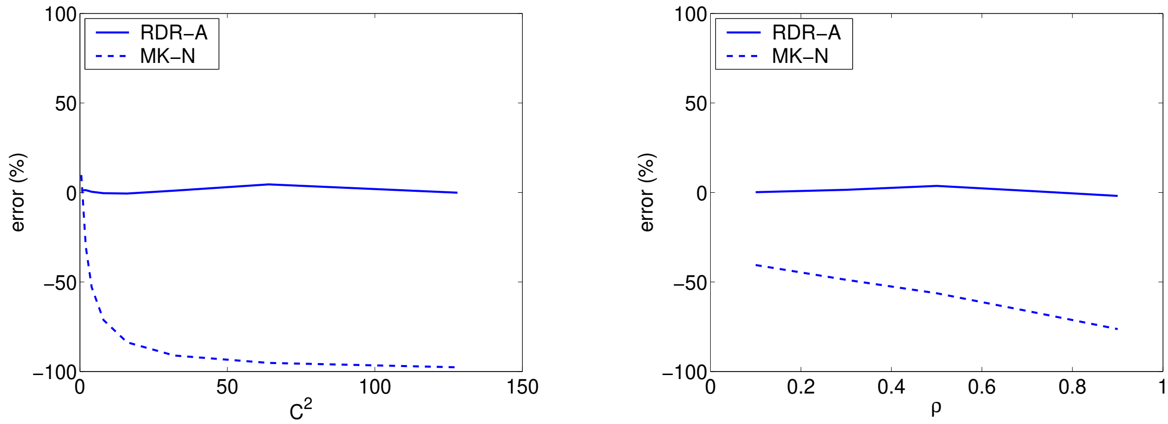


Figure 10: *Effect of aggregation. Graphs show error in mean delay of the 4th (lowest priority) class in the MK-N and RDR-A approximations for an M/PH/2 with SMART prioritization, as compared with simulation. On the left as a function of C^2 where $\rho = 0.8$, and on the right as a function of ρ where $C^2 = 8$. The classes all have a 2-phase PH distribution with the same squared coefficient of variation C^2 and different means: 1, 2, 4, and 8.*

the single server system, $\frac{40}{11}$ for the 2-server system, and $\frac{80}{11}$ for the 4-server system (recall that in a system with k servers, each server runs at $1/k$th the speed).

The most interesting observation is that more servers lead to less differentiation between SMART and STUPID schemes. For example, at load $\rho = 0.6$, there is a factor of five differentiation between SMART and STUPID with one server and only a 25% difference between SMART and STUPID with four servers. The effect appears more prominent under lighter load. This can be explained by recalling our earlier observation that multi-server systems allow short jobs a chance to jump ahead of long jobs, hence the negative effects of the STUPID scheme are mitigated.

4.3 How effective is class aggregation: RDR-A

In the early 80's Mitrani and King (later followed by Nishida in the early 90's) proposed analyzing prioritization in a multi-server system via aggregation as follows: To obtain the mean response time of the m^{th} class, simply aggregate classes 1 through $m-1$ into a single high-priority class, and let class m represent the low-priority class – then analyze the remaining two class system. The above MK-N approximation required further approximating the single aggregate class by an *exponential* job size distribution, since it was not known how to analyze even a two class multi-server system with non-exponential job size distributions.

Since RDR enables the analysis of multi-server priority queues with general PH job size distributions, we can reapply the MK-N aggregation idea, but where now we are able to capture the higher moments of the aggregated class. We call this approximation RDR-A, since it combines the use of RDR together with aggregation.

To understand the effect of aggregation, we consider a two server system with four priority classes. All the classes have a two phase PH distribution, with varying squared coefficient of variation (C^2). The classes differ however in their mean, having means 1, 2, 4, and 8, respectively, and are prioritized according to the SMART scheme; classes with lower means have higher priority. (STUPID prioritization yields similar insights.) Figure 10 examines the error in the mean delay of the 4th class under RDR-A and under MK-N as a function of C^2 (left) and as a function of ρ (right).

We see that the error in RDR-A is never more than 5% regardless of C^2 or ρ. By contrast, the error in MK-N is almost never less than 50%, and gets worse under higher load and C^2. We find experimentally that when the classes are identical, RDR-A incurrs only slightly more error than RDR. This makes sense since aggregating identical classes does not incur additional variability. However when the classes are different, as in the case of SMART scheduling in Figure 10, the error can increase to 5% under RDR-A as ρ and C^2 are varied, while it remains below 3% for RDR over the full range of ρ and C^2 depicted in Figure 10.

The bottom line is that "aggregation into two classes" is a good method for approximating prioritization in multi-server systems where the number of classes is $m > 2$. However, the aggregation needs to be done carefully – the distribution of the aggregate class must be modeled more closely than can be captured by an exponential distribution. Thus another benefit of RDR is revealed; by allowing for PH job size distributions it enables more accurate approximations of multi-class systems via aggregation.

5 Conclusion

This paper introduces the RDR technique, providing the first near-exact analysis of an M/PH/k queue with $m \geq 2$ priority classes. The RDR algorithm is efficient (requiring only a second or two for each data point in the paper) and accurate (resulting in $< 2\%$ error for all cases that we studied). Furthermore, RDR appears to maintain its accuracy across a wide range of loads and job size variability (in this paper we studied load ρ, ranging from 0.05 to 0.95 and studied squared coefficient of variation, C^2, ranging from 1 to 128).

Although the RDR algorithm is efficient when the number of priority classes is small, it becomes less practical when the number of priority classes grows (e.g., for an M/M/2 with 10 priority classes, the running time can get as high as tens of seconds). Hence we also introduce the RDR-A approximation, which works by aggregating the $m > 2$ priority classes into only two priority classes. The distribution of each aggregate class is then captured by a PH distribution, and the resulting 2 class system (with PH job sizes) is solved using RDR. The RDR-A algorithm is extremely efficient (< 1 second for a data point, regardless of the number of classes), since its running time is that of the RDR algorithm for only two classes. Furthermore, for the examples in this paper, the RDR-A algorithm has high accuracy ($< 5\%$ error) across a wide range of loads and C^2 values.

We use our analysis to obtain insights about priority queueing in multi-server systems. We start by comparing multi-server systems with single server systems of equal capacity. We find that the effect of prioritization in multi-server systems cannot be predicted by considering a comparable single server system. The reason is that adding servers creates complex effects not present in a single server. For example, multiple servers provide a strong benefit in dealing with highly variable job sizes, but they also hinder performance under lighter load. We also compare multi-server with single server systems, by evaluating the error in the Bondi-Buzen (BB) approximation which is based on relating multi-server performance under prioritization to single-server performance. We find that the error in BB can become large, when C^2 grows or the number of classes grows.

We next consider the effect of "smart" prioritization, where classes of jobs with smaller means are given priority over those with larger means. We find that "smart" prioritization has a much stronger effect in a single-server system, than in a multi-server system of equal capacity. This can be explained in part by the observation that multiple servers inherently aid short jobs by allowing them to jump ahead of long jobs.

Lastly, we contrast different methods of class aggregation, when used to approximately analyze a high number of classes. We find that aggregation when done carefully – by capturing three moments of the aggregated class – works surprisingly well, resulting in very low error. However, when the aggregate class is approximated only with respect to its first moment (by just an exponential distribution), aggregation can be very poor, resulting in error of well over 50%. The fact that RDR allows the first analysis of classes with PH job size distributions enables this good aggregation approximation.

In this paper we have focused on the problem of multi-server queues with $m > 2$ priorities. What makes this problem difficult is the fact that its Markov chain representation grows infinitely in m dimensions, and there are dependencies between those dimensions (the class M jobs depend on the class H jobs, and the class L jobs depend on both the class M and class H jobs). The RDR algorithm greatly simplifies this problem by reducing the dimensionality of the Markov chain to just one. There are many other problems that also exhibit high dimensionality in their Markov chain representation, and the RDR method introduced here may be applicable to those problems as well.

References

[1] D. Bertsimas and D. Nakazato. The distributional Little's Law and its applications. *Operations Research*, 43(2):298–310, 1995.

[2] A. Bondi and J. Buzen. The response times of priority classes under preemptive resume in M/G/m queues. In *ACM Sigmetrics*, pages 195–201, August 1984.

[3] L. Bright and P. Taylor. Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes. *Stochastic Models*, 11:497–514, 1995.

[4] J. Buzen and A. Bondi. The response times of priority classes under preemptive resume in M/M/m queues. *Operations Research*, 31:456–465, 1983.

[5] A. Cobham. Priority assignment in waiting line problems. *Operations Research*, 2:70–76, 1954.

[6] R. Davis. Waiting-time distribution of a multi-server, priority queueing system. *Operations Research*, 14:133–136, 1966.

[7] W. Feng, M. Kawada, and K. Adachi. Analysis of a multiserver queue with two priority classes and (M,N)-threshold service schedule ii: preemptive priority. *Asia-Pacific Journal of Operations Research*, 18:101–124, 2001.

[8] H. Gail, S. Hantler, and B. Taylor. Analysis of a non-preemptive priority multiserver queue. *Advances in Applied Probability*, 20:852–879, 1988.

[9] H. Gail, S. Hantler, and B. Taylor. On a preemptive Markovian queues with multiple servers and two priority classes. *Mathematics of Operations Research*, 17:365–391, 1992.

[10] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

[11] E. Kao and K. Narayanan. Modeling a multiprocessor system with preemptive priorities. *Management Science*, 2:185–97, 1991.

[12] E. Kao and S. Wilson. Analysis of nonpreemptive priority queues with multiple servers and two priority classes. *European Journal of Operational Research*, 118:181–193, 1999.

[13] E. P. C. Kao and K. S. Narayanan. Computing steady-state probabilities of a nonpreeptive priority multiserver queue. *Journal on Computing*, 2(3):211 – 218, 1990.

[14] A. Kapadia, M. Kazumi, and A. Mitchell. Analysis of a finite capacity nonpreemptive priority queue. *Computers and Operations Research*, 11:337–343, 1984.

[15] O. Kella and U. Yechiali. Waiting times in the non-preemptive priority M/M/c queue. *Stochastic Models*, 1:257 – 262, 1985.

[16] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM, 1999.

[17] H. Leemans. *The Two-Class Two-Server Queue with Nonpreemptive Heterogeneous Priority Structures*. PhD thesis, K.U.Leuven, 1998.

[18] D. McWherter, B. Schroeder, N. Ailamaki, and M. Harchol-Balter. Priority mechanisms for OLTP and transactional web applications. In *Proceedings of the 20th International Conference on Data Engineering (ICDE 2004)*, pages 535–546, April 2004.

[19] D. Miller. Steady-state algorithmic analysis of M/M/c two-priority queues with heterogeneous servers. In R. L. Disney and T. J. Ott, editors, *Applied probability - Computer science, The Interface, volume II*, pages 207–222. Birkhauser, 1992.

[20] I. Mitrani and P. King. Multiprocessor systems with preemptive priorities. *Performance Evaluation*, 1:118–125, 1981.

[21] M. Neuts. Moment formulas for the Markov renewal branching process. *Advances in Applied Probabilities*, 8:690–711, 1978.

[22] B. Ngo and H. Lee. Analysis of a pre-emptive priority M/M/c model with two types of customers and restriction. *Electronics Letters*, 26:1190–1192, 1990.

[23] T. Nishida. Approximate analysis for heterogeneous multiprocessor systems with priority jobs. *Performance Evaluation*, 15:77–88, 1992.

[24] T. Osogami. Analysis of a QBD process that depends on background QBD processes. Technical Report CMU-CS-04-163, School of Computer Science, Carnegie Mellon University, 2004. Implementation software available at www.cs.cmu.edu/∼osogami/code/index.html.

[25] T. Osogami and M. Harchol-Balter. A closed-form solution for mapping general distributions to minimal PH distributions. In *Performance TOOLS*, pages 200–217, 2003.

[26] T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. Analysis of cycle stealing with switching cost. In *ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 184 – 195, 2003.

[27] A. Sleptchenko. Multi-class, multi-server queues with non-preemptive priorities. Technical Report 2003-016, EURANDOM, Eindhoven University of Technology, 2003.

[28] A. Sleptchenko, A. van Harten, and M. van der Heijden. An exact solution for the state probabilities of the multi-class, multi-server queue with preemptive priorities, 2003 – Manuscript.

[29] W. Whitt. A review of $L = \lambda W$ and extensions. *Queueing Systems*, 9:235–268, 1991.

[30] A. Wierman, T. Osogami, M. Harchol-Balter, and A. Scheller-Wolf. How many servers are best in a dual-priority M/PH/k system. *Unpublished manuscript. In submission.*, 2005.

A Neuts' algorithm for computing moments of Inter-level passage times in QBD processes

Neuts' algorithm [21] is an efficient algorithm that calculates the moments of various types of passage times in very general processes, i.e. M/G/1 type semi-Markov processes. Because of its generality, however, the description of the algorithm in [21] is sophisticated, and thus non-trivial to understand or implement. Since Neuts' algorithm can be applied to the performance analysis of many computer and communication systems, it is a shame that it has not been used more frequently in the literature.

The purpose of this section is to make Neuts' algorithm more accessible by re-describing his algorithm restricted to the first three moments of inter-level passage times in QBD processes. In [21], the analysis is limited to homogeneous (level independent) processes. However, it is trivially extended to nonhomogeneous (level dependent) QBD process which we need in this paper. Therefore, we describe Neuts' algorithm for nonhomogeneous QBD processes that repeat after level $\hat{\ell}$. We omit all proofs, which are provided in detail in [21], and we instead focus on intuition and interpretation. We include everything needed to apply Neuts' algorithm within our solution framework, so that readers who wish to apply our methodology can do so.

In our paper, we consider a QBD process with state space $E = \{(i, \ell) | 1 \leq i \leq n_\ell, \ell \geq 0\}$, which has generator matrix $\mathbf{Q}$:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{L}^{(0)} & \mathbf{F}^{(0)} & & \\ \mathbf{B}^{(1)} & \mathbf{L}^{(1)} & \mathbf{F}^{(1)} & \\ & \mathbf{B}^{(2)} & \mathbf{L}^{(2)} & \mathbf{F}^{(2)} \\ & & \ddots & \ddots \end{pmatrix}$$

where $\mathbf{L}^{(\ell)}$ is a $n_\ell \times n_\ell$ matrix. We assume that our QBD process has a repeating structure: there exists $\hat{\ell} < \infty$ such that $\forall \ell \geq \hat{\ell}$, we have $\mathbf{B}^{(\ell)} = \mathbf{B}^{(\hat{\ell})}, \mathbf{L}^{(\ell)} = \mathbf{L}^{(\hat{\ell})}, \mathbf{F}^{(\ell)} = \mathbf{F}^{(\hat{\ell})}$.

We define level ℓ of the QBD as denoting the set of states of the form (i, ℓ) for $i = 1, ..., n_\ell$. Our goal can be roughly stated as deriving the passage time required to get from state (i, ℓ) to level $\ell - 1$ conditioned on the particular state first reached in level $\ell - 1$. More precisely, *we seek the first three moments of the distribution of the time required to get from state (i, ℓ) to state $(j, \ell - 1)$, given that state $(j, \ell - 1)$ is the first state reached in level $\ell - 1$* for any $\ell \geq 1$, and $1 \leq i \leq n_\ell$ and $1 \leq j \leq n_{\ell-1}$.

A.1 Outline

Our goal will be to derive the $n_\ell \times n_{\ell-1}$ matrix, $\mathbf{Z}_r^{(\ell)}$, where $(\mathbf{Z}_r^{(\ell)})_{ij}$ is the r^{th} moment of the distribution of time required to go from state (i, ℓ) to state $(j, \ell - 1)$, given that state $(j, \ell - 1)$ is the first state reached in level $\ell - 1$ when starting in (i, ℓ). Let

$$E_{ij}^{(\ell)} = \text{ Event that state } (j, \ell - 1) \text{ is the first state reached in level } \ell - 1 \text{ when starting in } (i, \ell).$$

Let

$$T_{ij}^{(\ell)} = \text{Time to go from state } (i,\ell) \text{ to state } (j,\ell-1).$$

Define the $n_\ell \times n_{\ell-1}$ probability matrix $\mathbf{G}^{(\ell)}$, where

$$(\mathbf{G}^{(\ell)})_{ij} = \Pr\left(E_{ij}^{(\ell)}\right).$$

The matrix $\mathbf{G}^{(\ell)}$ is a standard quantity in matrix analytic methods.

Now,

$$\begin{aligned}
(\mathbf{Z}_r^{(\ell)})_{ij} &= r^{\text{th}} \text{ moment of } T_{ij}^{(\ell)}, \text{ given event } E_{ij}^{(\ell)}. \\
&= \int_0^\infty x^r \Pr\left(T_{ij}^{(\ell)} = x \mid E_{ij}^{(\ell)}\right) dx \\
&= \int_0^\infty x^r \Pr\left(T_{ij}^{(\ell)} = x \text{ AND } E_{ij}^{(\ell)}\right) / \Pr\left(E_{ij}^{(\ell)}\right) dx \\
&= \frac{1}{(\mathbf{G}^{(\ell)})_{ij}} \int_0^\infty x^r \Pr\left(T_{ij}^{(\ell)} = x \text{ AND } E_{ij}^{(\ell)}\right) dx \\
&= \frac{(\mathbf{G}_r^{(\ell)})_{ij}}{(\mathbf{G}^{(\ell)})_{ij}}
\end{aligned}$$

where we have defined:

$$(\mathbf{G}_r^{(\ell)})_{ij} = \int_0^\infty x^r \Pr\left(T_{ij}^{(\ell)} = x \text{ AND } E_{ij}^{(\ell)}\right) dx = \int_0^\infty x^r \frac{d}{dx} \Pr\left(T_{ij}^{(\ell)} \le x \text{ AND } E_{ij}^{(\ell)}\right) dx.$$

Note that $(\mathbf{G}_r^{(\ell)})_{ij}$ is not a proper r^{th} moment in that its density functions doesn't integrate to 1, but rather to $(\mathbf{G}^{(\ell)})_{ij}$. The rest of the appendix is devoted to the questions of how to determine the two matrices: $\mathbf{G}_r^{(\ell)}$ and $\mathbf{G}^{(\ell)}$.

A.2 Preliminaries

This section presents a few preliminary derivations that we will need along the way. First, we will require a simple lemma regarding exponential random variables, which follows from their memoryless property.

Lemma 1 *Let X_1 and X_2 be independent exponential random variables. Let $W = min(X_1, X_2)$. Then*

$$\Pr(W < x \textit{ AND } X_1 < X_2) = \Pr(W < x) \cdot \Pr(X_1 < X_2).$$

We define the transition probability matrix, $\mathbf{P}(x)$, such that its (s,t) element, $\mathbf{P}(x)_{st}$, represents the probability that (i) the sojourn time at state s is $\leq x$ AND (ii) the first transition out of state s is to state t. By Lemma 1, $\mathbf{P}(x)_{st}$ is also the product of the probabilities of events (i) and (ii).

Matrix $\mathbf{P}(x)$ has the same structural shape as $\mathbf{Q}$, although its entries are clearly different, and thus it can be represented in terms of submatrices, analogous to those in $\mathbf{Q}$, indicating backward transitions between consecutive levels, local transitions within a level, and forward transitions between consecutive levels, as follows:

$$\mathbf{P}(x) = \begin{pmatrix} \boldsymbol{\mathcal{L}}^{(0)}(x) & \boldsymbol{\mathcal{F}}^{(0)}(x) & & \\ \boldsymbol{\mathcal{B}}^{(1)}(x) & \boldsymbol{\mathcal{L}}^{(1)}(x) & \boldsymbol{\mathcal{F}}^{(1)}(x) & \\ & \boldsymbol{\mathcal{B}}^{(2)}(x) & \boldsymbol{\mathcal{L}}^{(2)}(x) & \boldsymbol{\mathcal{F}}^{(2)}(x) \\ & & \ddots & \ddots \end{pmatrix}$$

Observe that $\boldsymbol{\mathcal{L}}^{(\ell)}(x)$ is an $n_\ell \times n_\ell$ submatrix for $\ell \geq 0$.

Next, we define the r-th moment of submatrices $\boldsymbol{\mathcal{B}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{L}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{F}}^{(\ell)}(x)$, as follows:

$$\boldsymbol{\mathcal{B}}_r^{(\ell)} = \int_0^\infty x^r \frac{d}{dx}\boldsymbol{\mathcal{B}}^{(\ell)}(x)dx; \qquad \boldsymbol{\mathcal{L}}_r^{(\ell)} = \int_0^\infty x^r \frac{d}{dx}\boldsymbol{\mathcal{L}}^{(\ell)}(x)dx; \qquad \boldsymbol{\mathcal{F}}_r^{(\ell)} = \int_0^\infty x^r \frac{d}{dx}\boldsymbol{\mathcal{F}}^{(\ell)}(x)dx$$

for $r = 1, 2, 3$, and $\ell \geq 0$, where an integral of a matrix $\mathbf{M}$ is a matrix of the integrals of the elements in $\mathbf{M}$.

We now define the limits as $x \to \infty$ of $\boldsymbol{\mathcal{B}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{L}}^{(\ell)}(x)$, $\boldsymbol{\mathcal{F}}^{(\ell)}(x)$, as follows:

$$\boldsymbol{\mathcal{B}}^{(\ell)} = \lim_{x\to\infty} \boldsymbol{\mathcal{B}}^{(\ell)}(x); \qquad \boldsymbol{\mathcal{L}}^{(\ell)} = \lim_{x\to\infty} \boldsymbol{\mathcal{L}}^{(\ell)}(x); \qquad \boldsymbol{\mathcal{F}}^{(\ell)} = \lim_{x\to\infty} \boldsymbol{\mathcal{F}}^{(\ell)}(x);$$

for $\ell \geq 0$.

If $\hat{\ell}$ denotes the level where the QBD starts to repeat, then we define:

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}}^{(\hat{\ell})} \quad \boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{L}}^{(\hat{\ell})} \quad \boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{F}}^{(\hat{\ell})}$$

and

$$\boldsymbol{\mathcal{B}}_r = \boldsymbol{\mathcal{B}}_r^{(\hat{\ell})} \quad \boldsymbol{\mathcal{L}}_r = \boldsymbol{\mathcal{L}}_r^{(\hat{\ell})} \quad \boldsymbol{\mathcal{F}}_r = \boldsymbol{\mathcal{F}}_r^{(\hat{\ell})}.$$

Example

Consider the QBD process shown in Figure 11.

Let $\gamma_1 = \lambda + \mu_1 + \alpha$ and $\gamma_2 = \lambda + \mu_2 + \beta$. Then, the corresponding transition probability matrices and their moments look as follows:

$$\boldsymbol{\mathcal{F}}^{(\ell)}(x) \quad = \quad \begin{pmatrix} 1 - e^{-\gamma_1 x} & 0 \\ 0 & 1 - e^{-\gamma_2 x} \end{pmatrix} \begin{pmatrix} \frac{\lambda}{\gamma_1} & 0 \\ 0 & \frac{\lambda}{\gamma_2} \end{pmatrix}$$

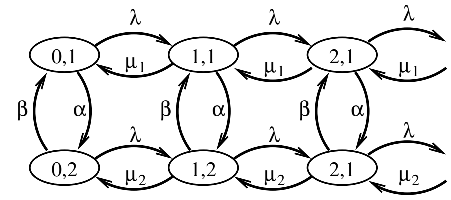


Figure 11: An example of a QBD process

$$
\begin{aligned}
\boldsymbol{\mathcal{L}}^{(\ell)}(x) &= \begin{pmatrix} 1-e^{-\gamma_1 x} & 0 \\ 0 & 1-e^{-\gamma_2 x} \end{pmatrix} \begin{pmatrix} 0 & \frac{\alpha}{\gamma_1} \\ \frac{\beta}{\gamma_2} & 0 \end{pmatrix} \\
\boldsymbol{\mathcal{B}}^{(\ell)}(x) &= \begin{pmatrix} 1-e^{-\gamma_1 x} & 0 \\ 0 & 1-e^{-\gamma_2 x} \end{pmatrix} \begin{pmatrix} \frac{\mu_1}{\gamma_1} & 0 \\ 0 & \frac{\mu_2}{\gamma_2} \end{pmatrix}
\end{aligned}
$$

and

$$
\boldsymbol{\mathcal{F}}^{(\ell)} = \begin{pmatrix} \frac{\lambda}{\gamma_1} & 0 \\ 0 & \frac{\lambda}{\gamma_2} \end{pmatrix} \qquad \boldsymbol{\mathcal{L}}^{(\ell)} = \begin{pmatrix} 0 & \frac{\alpha}{\gamma_1} \\ \frac{\beta}{\gamma_2} & 0 \end{pmatrix} \qquad \boldsymbol{\mathcal{B}}^{(\ell)} = \begin{pmatrix} \frac{\mu_1}{\gamma_1} & 0 \\ 0 & \frac{\mu_2}{\gamma_2} \end{pmatrix}
$$

and

$$
\begin{aligned}
\boldsymbol{\mathcal{F}}_r^{(\ell)} &= \begin{pmatrix} \frac{r!}{\gamma_1^r} & 0 \\ 0 & \frac{r!}{\gamma_2^r} \end{pmatrix} \begin{pmatrix} \frac{\lambda}{\gamma_1} & 0 \\ 0 & \frac{\lambda}{\gamma_2} \end{pmatrix} \\
\boldsymbol{\mathcal{L}}_r^{(\ell)} &= \begin{pmatrix} \frac{r!}{\gamma_1^r} & 0 \\ 0 & \frac{r!}{\gamma_2^r} \end{pmatrix} \begin{pmatrix} 0 & \frac{\alpha}{\gamma_1} \\ \frac{\beta}{\gamma_2} & 0 \end{pmatrix} \\
\boldsymbol{\mathcal{B}}_r^{(\ell)} &= \begin{pmatrix} \frac{r!}{\gamma_1^r} & 0 \\ 0 & \frac{r!}{\gamma_2^r} \end{pmatrix} \begin{pmatrix} \frac{\mu_1}{\gamma_1} & 0 \\ 0 & \frac{\mu_2}{\gamma_2} \end{pmatrix}
\end{aligned}
$$

for $\ell \geq 1$.

A.3 Where we're going

We will use the matrices introduced above to derive $\mathbf{G}^{(\ell)}$ and $\mathbf{G}_r^{(\ell)}$ for $r = 1, 2, 3$. In Section A.4, we derive the repeating part, where $\ell \geq \hat{\ell}$ for these matrices. Specifically we define

$$
\mathbf{G} = \mathbf{G}^{(\hat{\ell})} \quad \text{and} \quad \mathbf{G}_r = \mathbf{G}_r^{(\hat{\ell})}
$$

and proceed to derive $\mathbf{G}$ and $\mathbf{G}_r$. In Section A.5 we derive the nonrepeating part, $\mathbf{G}^{(\ell)}$ and $\mathbf{G}_r^{(\ell)}$ for $\ell < \hat{\ell}$. In deriving the non-repeating parts, we will make use of the repeating parts, $\mathbf{G}$ and $\mathbf{G}_r$, derived in Section A.4.

A.4 Moments of passage time in the repeating part

The quantity $\mathbf{G} = \mathbf{G}^{(\hat{\ell})}$ is a standard quantity in matrix analytic methods, and the reference [16] provides many methods by which this matrix can be computed. Below we describe the most straighforward (but slow) algorithm for generating $\mathbf{G}$. The idea is to simply iterate the following equation until it converges:

$$\mathbf{G} = \boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{L}}\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G} \tag{2}$$

The intuition behind Equation (2) should be clear: Recall that the $(i,j)^{th}$ entry of $\mathbf{G}^{(\ell)}$ represents the probability that state $(j, \ell-1)$ is the first state reached in level $\ell - 1$, given that we start in state (i, ℓ). Now the right hand side of Equation (2) represents this same probability by conditioning on whether the first move is a backwards transition, a local transition, or a forwards transition. More specifically, $\boldsymbol{\mathcal{B}}_{ij}^{(\ell)}$ represents the probability that the first move leaving state (i, ℓ) is to state $(j, \ell-1)$. The quantity, $\boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}^{(\ell)}$ represents the probability that the first transition out of state (i, ℓ) is to state (v, ℓ) for some v multiplied by the probability that the the first state reached in level $\ell - 1$ is $(j, \ell - 1)$ when we start in state (v, ℓ), summed over all possible $v = 1 \dots n_\ell$. Finally, the quantity, $\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)}$ represents the product of three terms: (i) the probability that the first transition from (i, ℓ) is to some state $(v_1, \ell+1)$ in level $\ell + 1$, (ii) the probability that from state $(v_1, \ell+1)$ the first state reached in level ℓ is some (v_2, ℓ), and (iii) the probability that from state (v_2, ℓ) the first state reached in level $\ell - 1$ is state $(j, \ell - 1)$, where the product is summed over all possible $v_1 = 1 \dots n_{\ell+1}$, $v_2 = 1 \dots n_\ell$. Since we are in the repeating region, all the superscripts are equal to $\hat{\ell}$, and can be dropped by our notation.

Matrices $\mathbf{G}_r$ for $r = 1, 2, 3$ are derived in a similar manner, by using $\mathbf{G}$ which we have already derived. Matrix $\mathbf{G}_1$ is obtained by iterating:

$$\mathbf{G}_1 = \boldsymbol{\mathcal{B}}_1 + \boldsymbol{\mathcal{L}}_1\mathbf{G} + \boldsymbol{\mathcal{L}}\mathbf{G}_1 + \boldsymbol{\mathcal{F}}_1\mathbf{G}\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}_1\mathbf{G} + \boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G}_1. \tag{3}$$

Similarly, matrix $\mathbf{G}_2$ is obtained by iterating:

$$\begin{aligned}\mathbf{G}_2 \;=\;& \boldsymbol{\mathcal{B}}_2 + \boldsymbol{\mathcal{L}}_2\mathbf{G} + 2\boldsymbol{\mathcal{L}}_1\mathbf{G}_1 + \boldsymbol{\mathcal{L}}\mathbf{G}_2 \\ & + \boldsymbol{\mathcal{F}}_2\mathbf{G}\mathbf{G} + 2\boldsymbol{\mathcal{F}}_1(\mathbf{G}_1\mathbf{G} + \mathbf{G}\mathbf{G}_1) + \boldsymbol{\mathcal{F}}(\mathbf{G}_2\mathbf{G} + 2\mathbf{G}_1\mathbf{G}_1 + \mathbf{G}\mathbf{G}_2)\end{aligned} \tag{4}$$

and matrix $\mathbf{G}_3$ is obtained by iterating:

$$\begin{aligned}\mathbf{G}_3 \;=\;& \boldsymbol{\mathcal{B}}_3 + \boldsymbol{\mathcal{L}}_3\mathbf{G} + 3\boldsymbol{\mathcal{L}}_2\mathbf{G}_1 + 3\boldsymbol{\mathcal{L}}_1\mathbf{G}_2 + \boldsymbol{\mathcal{L}}\mathbf{G}_3 \\ & + \boldsymbol{\mathcal{F}}_3\mathbf{G}\mathbf{G} + 3\boldsymbol{\mathcal{F}}_2(\mathbf{G}_1\mathbf{G} + \mathbf{G}\mathbf{G}_1) + 3\boldsymbol{\mathcal{F}}_1(\mathbf{G}_2\mathbf{G} + 2\mathbf{G}_1\mathbf{G}_1 + \mathbf{G}\mathbf{G}_2) \\ & + \boldsymbol{\mathcal{F}}(\mathbf{G}_3\mathbf{G} + 3\mathbf{G}_2\mathbf{G}_1 + 3\mathbf{G}_1\mathbf{G}_2 + \mathbf{G}\mathbf{G}_3).\end{aligned} \tag{5}$$

We now give intuition behind expressions (3)-(5). The right hand side of (3) can be divided into three parts: <u>*Part 0*</u>: $\boldsymbol{\mathcal{B}}_1$, <u>*Part 1*</u>: $\boldsymbol{\mathcal{L}}_1\mathbf{G}+\boldsymbol{\mathcal{L}}\mathbf{G}_1$, and <u>*Part 2*</u>: $\boldsymbol{\mathcal{F}}_1\mathbf{G}\mathbf{G}+\boldsymbol{\mathcal{F}}\mathbf{G}_1\mathbf{G}+\boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G}_1$. For $h=0,1,2$, the (i,j) element of *Part h* gives "the first moment of the distribution of $T_{ij}^{(\ell)}$ given that the first transition out of state (i,ℓ) is to level $\ell+h-1$ and $E_{ij}^{(\ell)}$" multiplied by "the probability that the first transition out of state (i,ℓ) is to level $\ell+h-1$ and $E_{ij}^{(\ell)}$." *Part 1* consists of two terms. The first term, $\boldsymbol{\mathcal{L}}_1\mathbf{G}$, is the contribution of the time to the first transition, and the second term, $\boldsymbol{\mathcal{L}}\mathbf{G}_1$, is the contribution of the time it takes to reach $(j,\ell-1)$ after the first transition. Similarly, *Part 2* consists of three terms. The first term, $\boldsymbol{\mathcal{F}}_1\mathbf{G}\mathbf{G}$, is the contribution of the time to the first transition, the second term, $\boldsymbol{\mathcal{F}}\mathbf{G}_1\mathbf{G}$, is the contribution of the time it takes to come back from level $\ell+1$ to level ℓ after the first transition, and the third term, $\boldsymbol{\mathcal{F}}\mathbf{G}\mathbf{G}_1$, is the contribution of the time it takes to go from level ℓ to level $\ell-1$.

The right hand sides of (4) and (5) can similarly be divided into three parts: *Part 0* consists of terms containing $\boldsymbol{\mathcal{B}}$ or $\boldsymbol{\mathcal{B}}_r$; *Part 1* consists of terms containing $\boldsymbol{\mathcal{L}}$ or $\boldsymbol{\mathcal{L}}_r$; *Part 2* consists of terms containing $\boldsymbol{\mathcal{F}}$ or $\boldsymbol{\mathcal{F}}_r$. The three parts of (4) and (5) can be interpreted exactly the same way as the three parts of (3) except that "the first moment" in (3) must be replaced by "the second moment" and "the third moment" in (4) and (5), respectively. The three terms in *Part 1* of (4) can be interpreted as follows. Let X be the time to the first transition and let Y be the time it takes from level ℓ to level $\ell-1$. Then, the second moment of the distribution of these two times is

$$E[(X+Y)^2]=E[(X)^2]+2E[X]E[Y]+E[(Y)^2],$$

since X and Y are independent. Roughly speaking, $\boldsymbol{\mathcal{L}}_2\mathbf{G}$ corresponds to $E[(X)^2]$, $2\boldsymbol{\mathcal{L}}_1\mathbf{G}_1$ corresponds to $2E[X]E[Y]$, and $\boldsymbol{\mathcal{L}}\mathbf{G}_2$ corresponds to $E[(Y)^2]$. The other terms can be interpreted in the same way.

A.5 Extension to nonrepeating part

For $\ell<\hat{\ell}$, $\mathbf{G}^{(\ell)}$ is calculated recursively as follows:

$$\begin{aligned}\mathbf{G}^{(\ell)} &= \boldsymbol{\mathcal{B}}^{(\ell)}+\boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}^{(\ell)}+\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)}\\ &= \left(\mathbf{I}-\boldsymbol{\mathcal{L}}^{(\ell)}-\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1}\boldsymbol{\mathcal{B}}^{(\ell)}.\end{aligned}$$

The intuition for the above formulation is the same as in the previous section.

For $\ell<\hat{\ell}$, $\mathbf{G}_r^{(\ell)}$ is calculated recursively:

$$\begin{aligned}\mathbf{G}_1^{(\ell)} &= \boldsymbol{\mathcal{B}}_1^{(\ell)}+\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}^{(\ell)}+\boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}_1^{(\ell)}+\boldsymbol{\mathcal{F}}_1^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)}+\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)}+\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}\\ &= \left(\mathbf{I}-\boldsymbol{\mathcal{L}}^{(\ell)}-\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1}\left(\boldsymbol{\mathcal{B}}_1^{(\ell)}+\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}^{(\ell)}+\boldsymbol{\mathcal{F}}_1^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)}+\boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)}\right)\end{aligned}$$

$$
\begin{aligned}
\mathbf{G}_2^{(\ell)} &= \boldsymbol{\mathcal{B}}_2^{(\ell)} + \boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_1^{(\ell)} + \boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}_2^{(\ell)} \\
&\quad + \boldsymbol{\mathcal{F}}_2^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) \\
&\quad + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_2^{(\ell)}) \\
&= \left(\mathbf{I} - \boldsymbol{\mathcal{L}}^{(\ell)} - \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1} \\
&\quad \Big(\boldsymbol{\mathcal{B}}_2^{(\ell)} + \boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_1^{(\ell)} \\
&\quad + \boldsymbol{\mathcal{F}}_2^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)})\Big)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{G}_3^{(\ell)} &= \boldsymbol{\mathcal{B}}_3^{(\ell)} + \boldsymbol{\mathcal{L}}_3^{(\ell)}\mathbf{G}^{(\ell)} + 3\boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}_1^{(\ell)} + 3\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_2^{(\ell)} + \boldsymbol{\mathcal{L}}^{(\ell)}\mathbf{G}_3^{(\ell)} + \boldsymbol{\mathcal{F}}_3^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} \\
&\quad + 3\boldsymbol{\mathcal{F}}_2^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) + 3\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_2^{(\ell)}) \\
&\quad + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_3^{(\ell+1)}\mathbf{G}^{(\ell)} + 3\mathbf{G}_2^{(\ell+1)}\mathbf{G}_1^{(\ell)} + 3\mathbf{G}_1^{(\ell+1)}\mathbf{G}_2^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_3^{(\ell)}). \\
&= \left(\mathbf{I} - \boldsymbol{\mathcal{L}}^{(\ell)} - \boldsymbol{\mathcal{F}}^{(\ell)}\mathbf{G}^{(\ell+1)}\right)^{-1} \\
&\quad \Big(\boldsymbol{\mathcal{B}}_3^{(\ell)} + \boldsymbol{\mathcal{L}}_3^{(\ell)}\mathbf{G}^{(\ell)} + 3\boldsymbol{\mathcal{L}}_2^{(\ell)}\mathbf{G}_1^{(\ell)} + 3\boldsymbol{\mathcal{L}}_1^{(\ell)}\mathbf{G}_2^{(\ell)} + \boldsymbol{\mathcal{F}}_3^{(\ell)}\mathbf{G}^{(\ell+1)}\mathbf{G}^{(\ell)} \\
&\quad + 3\boldsymbol{\mathcal{F}}_2^{(\ell)}(\mathbf{G}_1^{(\ell+1)}\mathbf{G}^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_1^{(\ell)}) + 3\boldsymbol{\mathcal{F}}_1^{(\ell)}(\mathbf{G}_2^{(\ell+1)}\mathbf{G}^{(\ell)} + 2\mathbf{G}_1^{(\ell+1)}\mathbf{G}_1^{(\ell)} + \mathbf{G}^{(\ell+1)}\mathbf{G}_2^{(\ell)}) \\
&\quad + \boldsymbol{\mathcal{F}}^{(\ell)}(\mathbf{G}_3^{(\ell+1)}\mathbf{G}^{(\ell)} + 3\mathbf{G}_2^{(\ell+1)}\mathbf{G}_1^{(\ell)} + 3\mathbf{G}_1^{(\ell+1)}\mathbf{G}_2^{(\ell)})\Big).
\end{aligned}
$$

A.6 Generalization allowed

Finally, we mention some generalizations that Neuts' algorithm allows. (1) We restricted ourselves to the first three moments, but this approach can be generalized to any higher moments. (2) We restricted ourselves to the first passage time from level ℓ to level $\ell-1$, but this can be generalized to the passage time from level ℓ to level $\ell-i$. (3) We restricted ourselves to QBD processes, but this can be generalized to M/G/1 type semi-Markov processes. (4) We restricted ourselves to the moments of the distribution of the duration of busy periods, but this can be generalized to the moments of the joint distribution of the duration of a busy period and the number of transitions during the busy period.