Exact Analysis of the M/M/k/setup Class of Markov Chains via Recursive Renewal Reward

Anshul Gandhi
School of Computer Science
Carnegie Mellon University
anshulg@cs.cmu.edu

Sherwin Doroudi
Tepper School of Business
Carnegie Mellon University
sdoroudi@andrew.cmu.edu

Mor Harchol-Balter
School of Computer Science
Carnegie Mellon University
harchol@cs.cmu.edu

Alan Scheller-Wolf
Tepper School of Business
Carnegie Mellon University
awolf@andrew.cmu.edu

ABSTRACT

The M/M/k/setup model, where there is a penalty for turning servers on, is common in data centers, call centers and manufacturing systems. Setup costs take the form of a time delay, and sometimes there is additionally a power penalty, as in the case of data centers. While the M/M/1/setup was exactly analyzed in 1964, no exact analysis exists to date for the M/M/k/setup with $k > 1$.

In this paper we provide the first exact, closed-form analysis for the M/M/k/setup and some of its important variants including systems in which idle servers delay for a period of time before turning off or can be put to sleep. Our analysis is made possible by our development of a new technique, Recursive Renewal Reward (RRR), for solving Markov chains with a repeating structure. RRR uses ideas from renewal reward theory and busy period analysis to obtain closed-form expressions for metrics of interest such as the transform of time in system and the transform of power consumed by the system. The simplicity, intuitiveness, and versatility of RRR makes it useful for analyzing Markov chains far beyond the M/M/k/setup. In general, RRR should be used to reduce the analysis of any 2-dimensional Markov chain which is infinite in at most one dimension and repeating to the problem of solving a system of polynomial equations. In the case where all transitions in the repeating portion of the Markov chain are skip-free and all up/down arrows are unidirectional, the resulting system of equations will yield a closed-form solution.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

Keywords

Queueing Theory; Performance; Resource Allocation

SIGMETRICS'13, June 17-21, 2013, Pittsburgh, PA, USA.

1. INTRODUCTION

Setup times (a.k.a. exceptional first service) are a fundamental component of computer systems and manufacturing systems, and therefore they have always played an important role in queueing theoretic analysis. In manufacturing systems it is very common for a job that finds a server idle to wait for the server to "warm up" before service is initiated. In retail and hospitals, the arrival of customers may necessitate bringing in an additional human server, which requires a setup time for the server to arrive. In computer systems, setup times are once again at the forefront of research, as they are the key issue in dynamic capacity provisioning for data centers.

In data centers, it is desirable to turn idle servers off, or reallocate the servers, to save power. This is because idle servers burn power at 60–70% of the peak rate, so leaving servers on and idle is wasteful [4]. Unfortunately, most companies are hesitant to turn off idle servers because the setup time needed to restart these servers is very costly; the typical setup times for servers is 200 seconds, while a job's service requirement is typically less than 1 second [16, 6]. Not only is the setup time prohibitive, but power is also burned at peak rate during the entire setup period, although the server is still not functional. Thus it is not at all obvious that turning off idle servers is advantageous.

Many ideas have been proposed to minimize the number of times that servers in a data center must undergo setup. One major line of research involves load prediction techniques [16, 21, 5, 12]. In the case where load is unpredictable, research has turned to looking at policies such as delayedoff, which delay turning off an idle server for some fixed amount of time, in anticipation of a new arrival [14, 11, 9]. Another line of research involves reducing setup times by developing low power sleep modes [11, 19].

Surprisingly, for all the importance of setup times, very little is known about their analysis. The M/G/1 with setup times was analyzed in 1964 by Welch [26]. The analysis of an M/M/k system with setup times, which we refer to as **M/M/k/setup**, however, has remained elusive, owing largely to the complexity of the underlying Markov chain. (Fig. 1 shows an M/M/k/setup with exponentially distributed setup times.) In 2010, various analytical approximations for the M/M/k/setup were proposed in [10]. These approximations work well provided that either load is low or the

setup time is low. The M/M/∞/setup was also analyzed in [10] and found to exhibit product form. Other than the above, no progress has been made on the M/M/k/setup. Even less is known about the **M/M/k/setup/delayedoff**, where idle servers delay for a finite amount of time before turning off, or the **M/M/k/setup/sleep**, where idle servers can either be turned off (high setup time, zero power) or put to sleep (lower setup time, low power). Section 3 describes these models in greater detail. Section 2 describes related prior work, including existing methods for solving general Markov chains with a repeating structure.

This paper is the first to derive an exact, closed-form solution for the M/M/k/setup, the M/M/k/setup/delayedoff, and the M/M/k/setup/sleep. We obtain the Laplace transform of response time, the z-transform of power consumption, and other important metrics for all of the above models.

Our solution is made possible by our development of a new technique for solving Markov chains with a repeating structure – Recursive Renewal Reward (RRR). RRR is based on using renewal reward theory to obtain the metrics of interest, while utilizing certain recursion theorems about the chain. Unlike matrix-analytic methods [17], RRR does not require finding the "rate" matrix. Another feature of RRR is that it is simple enough to be taught in an elementary stochastic processes course.

In general, RRR should be able to reduce the analysis of any 2-dimensional Markov chain which is finite in one dimension, say the vertical dimension, and infinite (with repeating structure) in the other (horizontal dimension) to the problem of solving a system of polynomial equations. Further, if in the repeating portion all horizontal transitions are skip-free and all vertical transitions are unidirectional, the resulting system of equations will be at most quadratic, yielding a closed-form solution (see Section 10 and Fig. 6 for more details). We thus anticipate that RRR will prove useful to other researchers in analyzing many new problems.

2. PRIOR WORK

The few papers that have looked at the M/M/k/setup are discussed in Section 1. For the M/M/k/setup/delayedoff, only iterative matrix-analytic approaches have been used [9]. No analysis exists for M/M/k/setup/sleep. We now discuss papers that have considered repeating Markov chains and have proposed techniques for solving these. We then comment on how these techniques might or might not apply to the M/M/k/setup.

2.1 Matrix-analytic based approaches

Matrix-analytic methods are a common approach for analyzing Markov chains with repeating structure. Such approaches are typically numerical, generally involving iteration to find the rate matrix, R. These approaches do not, in general, lead to closed forms or to any intuition, but are very useful for evaluating chains under different parameters.

There are cases where it is known that the R matrix can be stated explicitly [17]. This typically involves using a combinatorial interpretation for the R matrix. As described in [17], the class of chains for which the combinatorial view is tractable is narrow. However, in [25], the authors show that the combinatorial interpretation extends to a broader class of chains. Their class does not include the M/M/k/setup, however, which is more complicated because the transition (setup) rates are not independent of the number of jobs in system. Much research has been done on improving matrix-analytic methods to make the iteration faster. An example is [24], which develops a fast iterative procedure for finding the rate matrix for a broader class of chains than that in [25]. The authors in [24] also provide an explicit solution for the rate matrix in terms of infinite sums.

2.2 Generating function based approaches

Generating functions have also been applied to solve chains with a repeating structure. Like matrix-analytic methods these are not intuitive: Generating function approaches involve guessing the form of the solution and then solving for the coefficients of the guess, often leading to long computations. In theory, they can be used to solve very general chains (see for example [1]). We initially tried applying a generating function approach to the M/M/2/setup and found it to be incredibly complex and without intuition. This led us to seek a simpler and more intuitive approach.

2.3 M/M/k with vacations

Many papers have been written about the M/M/k system with vacations, see for example [28, 27, 23, 18]. While the Markov chain for the M/M/k with vacations looks similar to the M/M/k/setup, the dynamics of the two systems are very different. A server takes a vacation as soon as it is idle and there are no jobs in the queue. By contrast, a setup time is initiated by jobs arriving to the queue. In almost all of the papers involving vacations, the vacation model is severely restricted, allowing only a fixed group of servers to go on vacation at once. This is very different from our system in which any number of servers may be in setup at any time. The model in [18] comes closest to our model, although the authors use generating functions and assume that *all* idle servers are on vacation, rather than one server being in setup for each job in queue, which makes the transitions in their chain independent of the number of jobs.

2.4 Restricted models of M/M/k with setup

There have been a few papers [2, 3, 10] that consider a very restricted version of the M/M/k/setup, wherein at most one server can be in setup at a time. There has also been prior work [20] that considers an M/M/k system wherein a fixed subset of servers can be turned on and off based on load. The underlying Markov chains for all of these restricted systems are analytically tractable and lead to very simple closed-form expressions, since the rate at which servers turn on is always fixed. Our M/M/k/setup system is more general, allowing any number of servers to be in setup. This makes our problem much more challenging.

2.5 How our work differs from all of the above

To the best of our knowledge, we are the first to derive exact closed-form results for the M/M/k/setup problem, with $k > 1$. Our solution was made possible by our new RRR technique. RRR results in exact solutions, does not require any iteration, and does not involve infinite sums. Importantly, RRR is highly intuitive and very easy to apply. Using RRR, we go much further than the M/M/k setup, deriving exact closed-form results for important variants such as the M/M/k/setup/delayedoff and the M/M/k with multiple types of setups, neither of which has been solved analytically.

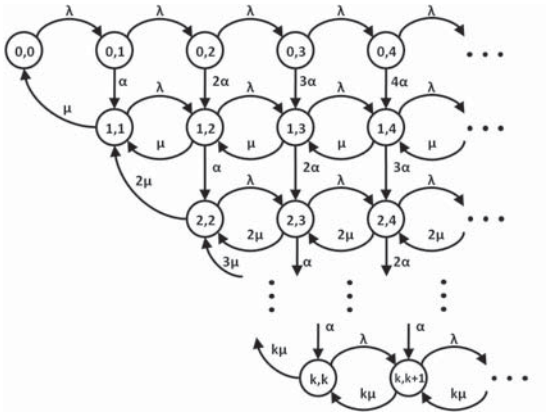


Figure 1: ***M/M/k/setup Markov chain. Each state is denoted by the pair*** (i, j)***, where*** i ***is the number of on servers, and*** j ***is the number of jobs in the system. The number of servers in setup is*** $\min\{j - i, k - i\}$***.***

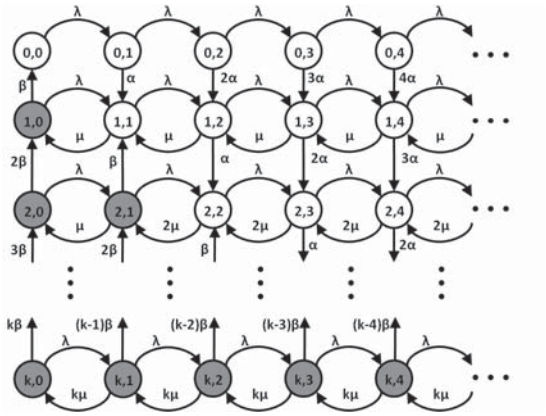


Figure 2: ***M/M/k/setup/delayedoff Markov chain. Each state is denoted by the pair*** (i, j)***, where*** i ***is the number of on or idle servers, and*** j ***is the number of jobs in the system. If*** $i < j$***, then the number of servers in setup is*** $\min\{j - i, k - i\}$***, and there are no idle servers. If*** $i > j$ ***(gray shaded states), the number of idle servers is*** $(i - j)$***, and there are no servers in setup. If*** $i = j$***, no servers are idle or in setup.***

3. MODEL

In our model jobs arrive according to a Poisson process with rate λ and are served at rate $\mu = \frac{1}{E[S]}$, where S denotes the job size and is exponentially distributed. For stability, we assume that $k \cdot \mu > \lambda$, where k is the number of servers in the system.

3.1 M/M/k/setup

In the M/M/k/setup system, each of the k servers is in one of three states: **off**, **on** (being used to serve a job), or **setup**. When a server is on or in setup, it consumes peak power of P_{peak} watts. When a server is off, it consumes zero power. Thus, when servers are not in use, they are immediately turned off to save power. Every arriving job that comes into the system picks an off server, if one exists, and puts it into setup mode; the job then joins the queue. We use I to denote the setup times, with $E[I] = \frac{1}{\alpha}$. Unless stated otherwise, we assume that setup times are exponentially distributed. When a job completes service at a server, say server s_1, and there are no remaining jobs left in the queue, then server s_1 is immediately turned off. However, if the queue is not empty, then server s_1 is not turned off, and the job at the head of the queue is directed to server s_1. Note that if the job at the head of the queue was already waiting on another server, say server s_2, in setup mode, the job at the head of the queue is still directed to server s_1. At this point, if there is a job in the queue that did not setup an off server on arrival (because there were no off servers), then server s_2 continues to be in setup for this job. If no such job exists in the queue, then server s_2 is turned off.

The Markov chain for the M/M/k/setup system is shown in Fig. 1. Each state is denoted by the pair (i, j), where i is the number of on servers, and j is the number of jobs in the system. Thus, the number of servers in setup is $\min\{j-i, k-i\}$. Note that the Markov chain is infinite in one dimension.

3.2 M/M/k/setup/delayedoff

The M/M/k/setup/delayedoff system is the same as the M/M/k/setup system, except that idle servers are not immediately turned off. Specifically, when a job completes service at a server, say server s_1, and there are no remaining jobs in the queue, s_1 remains waiting in the idle state for an exponentially distributed amount of time with mean $t_{wait} = \frac{1}{\beta}$. If a new job arrives while server s_1 is waiting, the job is immediately directed to s_1, which is already on. However, if no jobs arrive during server s_1's waiting period, then server s_1 is turned off. Intuitively, a higher t_{wait} results in lower response time, since servers are on longer, but may also increase power usage, since idle servers consume significant power.

The Markov chain for the M/M/k/setup/delayedoff system is shown in Fig. 2. The chain is the same as that for M/M/k/setup, except for the new gray shaded states which represent states with idle servers. As before, each state is denoted by the pair (i, j), where i is the number of on or idle servers, and j is the number of jobs in the system. For the M/M/k/setup/delayedoff system, each server can be in one of four states: off, on (busy), idle, or setup. If $i < j$, then the number of servers in setup is $\min\{j - i, k - i\}$, and there are no idle servers. If $i > j$ (gray shaded states), the number of idle servers is $(i - j)$, and there are no servers in setup. If $i = j$, no servers are idle or in setup.

3.3 M/M/k/setup/sleep

The M/M/k/setup/sleep is motivated by servers with sleep modes [11, 19], which allow an idle server to either be turned off or put to sleep. When a server is turned off, it consumes zero power. However, turning on an off server requires an exponentially distributed setup time, with rate α. By contrast, when a server is sleeping, it consumes some non-zero power, P_{sleep} watts, which is usually much smaller than the idle power, P_{idle} watts [11, 19]. When a sleeping server is turned on, it requires an exponentially distributed setup time, with rate $\omega > \alpha$. Thus, there is a tradeoff between turning off an idle server vs putting it to sleep.

One simple idea that leverages sleep states is to designate some subset of the k servers, say the first s servers, to "sleep" when idle, whereas the remaining $(k - s)$ servers are turned off when idle. An interesting question is what is a good value of s. To answer this question we introduce the M/M/k/setup/sleep model, which is the same as the

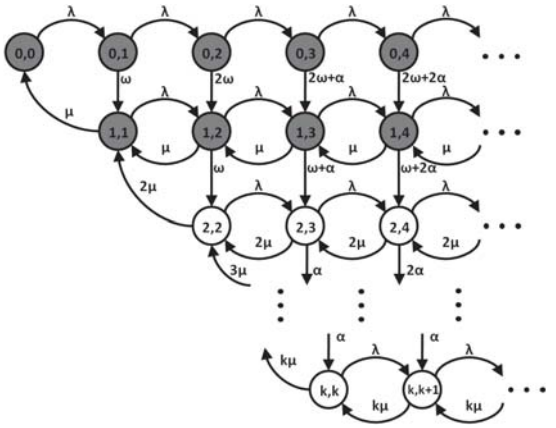


Figure 3: ***M/M/k/setup/sleep Markov chain. Each state is denoted by the pair*** (i, j)***, where*** i ***is the number of on servers, and*** j ***is the number of jobs in the system. The number of servers in fast setup is*** s***, which in this case is*** $s = 2$***. The number of servers in setup is*** $\min\{j - i, k - i\}$***. If*** $i < s$ ***(gray shaded states), the first*** $(s - i)$ ***servers setting up have a fast setup rate,*** ω***, while the other servers in setup have a slow setup rate,*** α***.***

M/M/k/setup, except that $s \leq k$ servers have a fast setup rate of ω and $(k-s)$ servers have a slow setup rate of α (see Fig. 3). As we will see later, the tradeoff between mean response time and mean power is highly sensitive to the choice of s (see Figs. 7(c) and 7(d)).

For ease of analysis, we make the following assumptions about the M/M/k/setup/sleep model: (i) In any group of servers in setup, we assume that the servers that have a fast setup rate (ω) complete setting up first. Thus, if we are in state (i, j) with $i < s$ (gray shaded states in Fig. 3), the first $(s - i)$ servers in setup will have a fast setup rate. Note that the i servers already on in state (i, j), with $i \leq s$, are those that had a fast setup rate. Thus, when we have $i \leq s$ servers busy, and a server is no longer in use, we put the server to sleep (as opposed to turning it off). (ii) If we have $i > s$ servers busy, and a server is no longer in use, we turn the server off (as opposed to putting it to sleep). This assumption allows us to save a lot of power when load goes down since off servers consume zero power. The above two assumptions are primarily for tractability of the M/M/k/setup/sleep Markov chain. In practice, ω is significantly higher than α. In this regime we simulated an M/M/k/setup/sleep system with and without the above two assumptions and found the results to be qualitatively unchanged.

4. THE RECURSIVE RENEWAL REWARD TECHNIQUE

In this section we provide a high-level description of our new Recursive Renewal Reward (RRR) technique, which yields exact, closed-form solutions for a range of Markov chains, including the M/M/k/setup (see Sections 5, 6 and 7), the M/M/k/setup/delayedoff (see Section 8) and the M/M/k/setup/sleep (see Section 9).

The RRR technique works by deriving the *expected "reward" earned per unit time in a Markov chain*, where the reward could be any quantity of interest. In the context of our M/M/k/setup problem, the reward earned at time t, $R(t)$, could be the number of jobs in system at time t, the square of the number of jobs in system, the current power usage, the number of servers that are on, or any other reward that can be expressed as a function of the state of the Markov chain.

To analyze the average rate of earning reward, we designate a *renewal state*, say $(0, 0)$,[1] which we call the *home state*, and then consider a *renewal cycle* to be the process of moving from the home state back to the home state. By renewal-reward theory, the average rate of earning reward is the same as the mean reward earned over a renewal cycle, which we denote by $\mathcal{R}$, divided by the mean length of the renewal cycle, denoted by $\mathcal{T}$.

$$\text{Average rate of earning} = \frac{\mathcal{R}}{\mathcal{T}} = \frac{E\left[\int_{\text{cycle}} R(t)dt\right]}{E\left[\int_{\text{cycle}} 1dt\right]}$$

For example, if the goal is to find the mean number of jobs, $E[N]$, for our chain, we simply define $R(t)$ to be the number of jobs at time t, which can be obtained from the state of the Markov chain at time t.

It turns out that the quantities $\mathcal{T}$ and $\mathcal{R}$ are very easy to compute! Consider a Markov chain, such as that in Fig. 4 or Fig. 5. The *repeating portion* of the chain is shown in gray. There are a finite number of *border states* which sit at the edge of the repeating chain and are colored black. We will see that computing $\mathcal{T}$ and $\mathcal{R}$ basically reduces to writing one equation for each border state[2]. For the case of $\mathcal{T}$, we will need the mean time to move one step left from each border state. For the case of $\mathcal{R}$, we will need the mean reward earned when moving one step left from each border state. Computing these border state quantities is made very easy via some neat recursion theorems. We demonstrate this process in the examples below. There are a few details which we will defer until after these examples. For instance, in general, it is necessary to also add equations for the non-repeating portion of the Markov chain. See Sections 7 and 10 for more details on the RRR technique.

5. M/M/1/SETUP

In this section we illustrate the RRR technique by applying it to the simple M/M/1/setup system, whose Markov chain is shown in Fig. 4. Here, the state of the system is represented as (i, j), where $i \in \{0, 1\}$ is the number of servers on and $0 \leq j < \infty$ is the number of jobs in the system. In general, i represents the *depth* (or row number) of the state, and j represents the *level* (or column number) of the state. We start by deriving $E[N]$, the mean number of jobs, and then move to more complex metrics. We choose the renewal state to be $(0, 0)$ and we define the reward earned at time t, $R(t)$, to be $N(t)$, the number of jobs in the system at time t. As explained in Section 4, all we need is $\mathcal{T}$ and $\mathcal{R}$.

5.1 Deriving $\mathcal{T}$ via $\mathbf{T^L_{0,1}}$ and $\mathbf{T^L_{1,1}}$

[1] In principle any state can be chosen as the renewal state, but some states allow for an easier (or shorter) analysis.

[2] Several techniques in the literature such as matrix-analytic methods [17] and stochastic complementation [22] also deal with border states, although none of them involve renewal-reward theory.

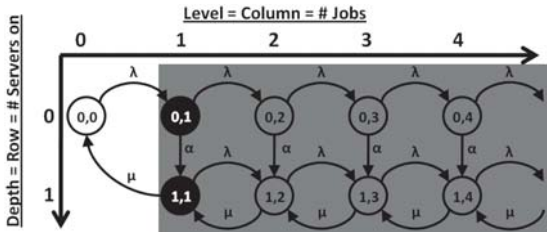


Figure 4: ***M/M/1/setup Markov chain with the repeating portion highlighted in gray and the border states shaded black.***

$\mathcal{T}$ is the mean time to get from our home state $(0,0)$ back to $(0,0)$. This can be viewed as $\frac{1}{\lambda}$, the mean time until we leave $(0,0)$ (which takes us to $(0,1)$) plus the mean time to get home from $(0,1)$. We make the further observation that the mean time to get home from $(0,1)$ is equal to $T^L_{0,1}$ (using notation from Table 1), the mean time to move left one level from $(0,1)$ (since moving left can only put us in $(0,0)$). We thus have:

$$\mathcal{T} = \frac{1}{\lambda} + T^L_{0,1} \tag{1}$$

We now need an equation for $T^L_{0,1}$ for the border state $(0,1)$, which will require looking at the other border state, $(1,1)$, as well. Starting with border state $(1,1)$, it is clear that $T^L_{1,1}$ is simply the mean length of an M/M/1 busy period, B_1. Thus, we have:

$$T^L_{1,1} = B_1 = \frac{1}{\mu - \lambda} \tag{2}$$

$T^L_{0,1}$ involves waiting in state $(0,1)$ for expected time $\frac{1}{\alpha+\lambda}$, before conditioning on where we transition to next. If we go to state $(1,1)$ we need an additional $T^L_{1,1}$. However if we go to state $(0,2)$ we need to add on the time to move one step left from $(0,2)$ (which by Fig. 4 takes us to $(1,1)$) and then an additional $T^L_{1,1}$. That is:

$$T^L_{0,1} = \frac{1}{\lambda+\alpha} + \frac{\alpha}{\lambda+\alpha} \cdot T^L_{1,1} + \frac{\lambda}{\lambda+\alpha}\left(T^L_{0,2} + T^L_{1,1}\right) \tag{3}$$

It is now time to invoke one of our recursion theorems, which holds for any M/M/k/setup chain:

THEOREM 1 (RECURSION THEOREM FOR MEAN TIME)
For the M/M/k/setup, the mean time to move one step left from state (i,j), $T^L_{i,j}$, is the same for all $j \geq k$.

Thm. 1 follows from the fact that the repeating portion of the Markov chain is identical for all states in a given row. The full proof of Thm. 1 (along with the proofs of all other theorems) is presented in Appendix A.

Using Thm. 1, we replace $T^L_{0,2}$ in Eq. (3) with $T^L_{0,1}$ to get:

$$T^L_{0,1} = \frac{1}{\lambda+\alpha} + \frac{\alpha}{\lambda+\alpha} \cdot T^L_{1,1} + \frac{\lambda}{\lambda+\alpha}\left(T^L_{0,1} + T^L_{1,1}\right) \tag{4}$$

Finally, noting that $T^L_{1,1} = B_1$ from Eq. (2), we have that:

$$T^L_{0,1} = \frac{1}{\lambda+\alpha} + \frac{\alpha}{\lambda+\alpha} \cdot B_1 + \frac{\lambda}{\lambda+\alpha}\left(T^L_{0,1} + B_1\right)$$

$$\implies T^L_{0j} = T^L_{0,1} = \frac{1 + (\lambda+\alpha)B_1}{\alpha} \tag{5}$$

Variable	Description
$\mathcal{T}$	Mean length of the renewal cycle
$\mathcal{R}$	Mean reward earned during a renewal cycle
$T^L_{i,j}$	Mean time until we first move one level left of (i,j), starting from (i,j)
$R^L_{i,j}$	Mean reward earned until we first move one level left of (i,j), starting from (i,j)
$p^L_{i\to d}$	Probability that after we first move one level left from state (i,j), we are at depth d
B_k	Mean length of an $M/M/k$ busy period

Table 1: ***Variables used in our analysis of*** $E[N]$.

Substituting $T^L_{0,1}$ from above into Eq. (1) gives us $\mathcal{T}$:

$$\mathcal{T} = \frac{\mu(\lambda+\alpha)}{\lambda\alpha(\mu-\lambda)} \tag{6}$$

5.2 Deriving $\mathcal{R}$ via $\mathbf{R^L_{0,1}}$ and $\mathbf{R^L_{1,1}}$

$\mathcal{R}$ denotes the reward earned in moving from $(0,0)$ back to $(0,0)$. Observing that we earn 0 reward in state $(0,0)$ (because there are no jobs in the system in that state), and observing that from state $(0,0)$ we can only next move to $(0,1)$, we have (using notation from Table 1):

$$\mathcal{R} \;=\; R^L_{0,1} \tag{7}$$

It now remains to compute the reward earned in moving one step left from $(0,1)$, which will require looking at the other border state, $(1,1)$, as well.

To do this, we invoke another recursion theorem, which again holds for any M/M/k/setup system:

THEOREM 2 (RECURSION THEOREM FOR MEAN REWARD)
For the M/M/k/setup, the mean reward earned in moving one step left from state $(i,j+1)$, $R^L_{i,j+1}$, satisfies $R^L_{i,j+1} = R^L_{i,j} + T^L_{i,j}$ for all $j \geq k$, where the reward tracks the number of jobs in the system.

Applying Thm. 2 to the Markov chain shown in Fig. 4, we have:

$$\begin{aligned} R^L_{1,1} &= \frac{1}{\lambda+\mu} \cdot 1 + \frac{\mu}{\lambda+\mu} \cdot 0 + \frac{\lambda}{\lambda+\mu}\left(R^L_{1,2} + R^L_{1,1}\right) \\ &= \frac{1}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu}\left((R^L_{1,1} + T^L_{1,1}) + R^L_{1,1}\right) \\ &= \frac{1}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu}\left((R^L_{1,1} + B_1) + R^L_{1,1}\right) \end{aligned} \tag{8}$$

(from Eq. (2))

$$\implies R^L_{1,1} = \frac{1+\lambda B_1}{\mu-\lambda} \tag{9}$$

Variable	Description
$\dot{\mathcal{R}}$	Mean reward earned (for z-transform) during a renewal cycle
$\dot{\mathcal{E}}$	Mean reward earned (for transform of power) during a renewal cycle
$\dot{R}^L_{i,j}$	Mean reward earned (for z-transform) until we first move one level left of (i,j), starting from (i,j)
$\dot{E}^L_{i,j}$	Mean reward earned (for z-transform of power) until we first move one level left of (i,j), starting from (i,j)

Table 2: ***Variables used in our transform analyses.***

Similarly, for border state $(0,1)$, we have:

$$R^L_{0,1} = \frac{1}{\lambda+\alpha}\cdot 1 + \frac{\alpha}{\lambda+\alpha}\cdot R^L_{1,1} + \frac{\lambda}{\lambda+\alpha}\left(R^L_{0,2} + R^L_{1,1}\right)$$

$$= \frac{1}{\lambda+\alpha} + \frac{\alpha}{\lambda+\alpha}\cdot R^L_{1,1}$$

$$+ \frac{\lambda}{\lambda+\alpha}\left((R^L_{0,1} + T^L_{0,1}) + R^L_{1,1}\right) \quad \text{(from Thm. 2)}$$

$$\implies R^L_{0,1} = \frac{1+\lambda T^L_{0,1} + (\lambda+\alpha)R^L_{1,1}}{\alpha}. \tag{10}$$

Substituting $R^L_{0,1}$ from above into Eq. (7) gives us $\mathcal{R}$:

$$\mathcal{R} = \frac{\mu(\lambda+\alpha)(\mu-\lambda+\alpha)}{\alpha^2(\mu-\lambda)^2} \tag{11}$$

5.3 Deriving E[N]

Since $E[N] = \frac{\mathcal{R}}{\mathcal{T}}$, combining Eq. (6) and Eq. (11), we get:

$$E[N] = \frac{\mathcal{R}}{\mathcal{T}} = \frac{\lambda}{\alpha} + \frac{\lambda}{\mu-\lambda} \tag{12}$$

The second term in the right hand side of Eq. (12) can be identified [15] as the mean number of jobs in an M/M/1 system (without setup). Thus, Eq. (12) is consistent with the known decomposition property for the M/M/1/setup system [26].

5.4 Deriving $\widehat{N}(z)$ and $\widetilde{T}(s)$

Deriving the z-transform of the number of jobs, $\widehat{N}(z) = E[z^N]$, is just as easy as deriving $E[N]$. The only difference is that our reward function is now $R(t) = z^{N(t)}$, where $N(t)$ is again the number of jobs in the system at time t. Thus

$$\widehat{N}(z) = E[z^N] = \frac{\dot{\mathcal{R}}}{\mathcal{T}},$$

where $\dot{\mathcal{R}} = E\left[\int_{\text{cycle}} z^{N(t)}dt\right]$ and $\mathcal{T}$ is the same as before.

We will again invoke a recursion theorem which applies to any M/M/k/setup (using notation from Table 2):

THEOREM 3 (RECURSION THEOREM FOR TRANSFORM OF REWARD) *For the M/M/k/setup, $\dot{R}^L_{i,j+1} = z\cdot\dot{R}^L_{i,j}$, for all $j \geq k$, where $\dot{R}$ tracks the z-transform of the number of jobs in the system.*

Let us now express $\dot{\mathcal{R}}$ by conditioning on the first step from $(0,0)$:

$$\dot{\mathcal{R}} = \frac{1}{\lambda} + \dot{R}^L_{0,1} \tag{13}$$

We again need one equation per border state:

$$\dot{R}^L_{1,1} = \frac{1}{\lambda+\mu}\cdot z + \frac{\lambda}{\lambda+\mu}\left(z\cdot\dot{R}^L_{1,1} + \dot{R}^L_{1,1}\right)$$

$$\dot{R}^L_{0,1} = \frac{1}{\lambda+\alpha}\cdot z + \frac{\alpha}{\lambda+\alpha}\cdot\dot{R}^L_{1,1} + \frac{\lambda}{\lambda+\alpha}\left(z\cdot\dot{R}^L_{0,1} + \dot{R}^L_{1,1}\right)$$

Solving the above system and substituting $\dot{R}^L_{0,1}$ into Eq. (13) allows us to express $\dot{\mathcal{R}}$ in closed form. This gives us $\widehat{N}(z)$, after some algebra, as follows:

$$\widehat{N}(z) = E[z^N] = \frac{\dot{\mathcal{R}}}{\mathcal{T}} = \frac{\alpha(\mu-\lambda)}{(\mu-\lambda z)(\alpha+\lambda-\lambda z)} \tag{14}$$

To get the Laplace transform of response time, $\widetilde{T}(s)$, we use the distributional Little's Law [13] (since M/M/1/setup is a First-In-First-Out system):

$$\widetilde{T}(s) = \widehat{N}\left(1-\frac{s}{\lambda}\right) = \frac{\alpha(\mu-\lambda)}{(s+\alpha)(\mu+s-\lambda)} \tag{15}$$

5.5 Deriving $\widehat{P}(z)$

We now derive $\widehat{P}(z)$, the z-transform of the power consumed for the M/M/1/setup. The server consumes zero power when it is off, but consumes peak power, P_{peak} watts, when it is on or in setup. This time, the reward is simply the transform of the energy consumed over the renewal cycle, $\dot{\mathcal{E}} = E\left[\int_{\text{cycle}} z^{P(t)}dt\right]$, where $P(t)$ is the power consumed at time t. We begin with the recursive theorem for $\dot{E}^L_{i,j}$, just like we had Thm. 3 for $\dot{R}^L_{i,j}$.

THEOREM 4 (RECURSION THEOREM FOR TRANSFORM OF POWER) *For the M/M/k/setup, $\dot{E}^L_{i,j+1} = \dot{E}^L_{i,j} = T^L_{i,j}\cdot z^{k\cdot P_{peak}}$, for all $j \geq k$.*

Thm. 4 gives us $\dot{E}^L_{i,j}$ in closed form, in terms of $T^L_{i,j}$. Following the usual renewal-reward approach, we get:

$$\widehat{P}(z) = E[z^P] = \frac{\dot{\mathcal{E}}}{\mathcal{T}} = \frac{\alpha(\mu-\lambda) + \lambda(\mu+\alpha)z^{P_{peak}}}{\mu(\lambda+\alpha)} \tag{16}$$

6. M/M/2/SETUP

The M/M/2/setup chain shown in Fig. 5 is analyzed similarly to the M/M/1/setup, except that there are now three border states, $(0,2)$, $(1,2)$, and $(2,2)$. The only complication is that when moving one level left from a given state, the resulting row is non-deterministic. For example, when moving left from $(1,3)$ in Fig. 5, we may end up in row 1 at $(1,2)$ or row 2 at $(2,2)$. We use $p^L_{i\to d}$ to denote the probability that once we move one level left from (i,j), we will be at depth d.[2] The following theorem proves that $p^L_{i\to d}$ is independent of j for all states (i,j) in the repeating portion.

THEOREM 5 (RECURSION THEOREM FOR PROBABILITY) *For the M/M/k/setup, for each $0 \leq d \leq k$ and for each $0 \leq i \leq k$, $p^L_{i\to d}$ is the same for all $j \geq k$.*

Thus, it suffices to compute $p^L_{i\to d}$ for the border states. These probabilities are used in Section 6.2.

6.1 Deriving $p^L_{i\to d}$

Solving for the $p^L_{i\to d}$ is easiest "bottom-up" (starting from the greatest depth, i). For $i = 2$, we have $p^L_{2\to 2} = 1$ for all $j > 2$, since we stay at depth 2 after moving left. For $i = 1$

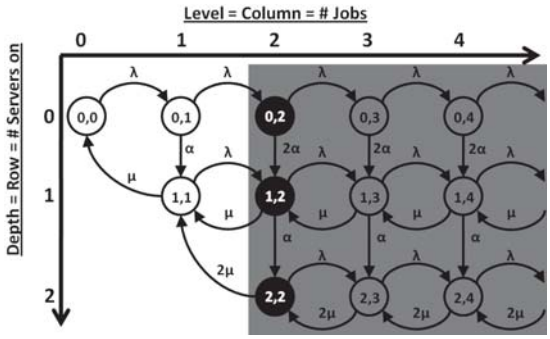


Figure 5: ***M/M/2/setup Markov chain with the repeating portion highlighted in gray and the border states shaded black.***

and $i = 0$, we follow the same approach of conditioning on the first step and using recursion theorems:

$$p^L_{1\to1} = \frac{\mu}{\lambda+\mu+\alpha} + \frac{\lambda}{\lambda+\mu+\alpha}\left(p^L_{1\to1}\right)^2 \tag{17}$$

$$p^L_{0\to1} = \frac{2\alpha}{\lambda+2\alpha}\left(p^L_{1\to1}\right) + \frac{\lambda}{\lambda+2\alpha}\left(p^L_{0\to1}\right)\left(p^L_{1\to1}\right) \tag{18}$$

Eqs. (17) and (18) can now be solved in closed form since they are of degree at most 2. Note that $p^L_{1\to2} = 1 - p^L_{1\to1}$ and $p^L_{0\to2} = 1 - p^L_{0\to1}$.

6.2 Deriving $\widehat{\mathbf{N}}(\mathbf{z})$ via $\dot{\mathbf{R}}^{\mathbf{L}}_{\mathbf{0,2}}$, $\dot{\mathbf{R}}^{\mathbf{L}}_{\mathbf{1,2}}$, and $\dot{\mathbf{R}}^{\mathbf{L}}_{\mathbf{2,2}}$

To derive $\widehat{N}(z) = E[z^N]$, we again need to find $\dot{\mathcal{R}}$ and $\mathcal{T}$, where $\dot{\mathcal{R}} = E\left[\int_{\text{cycle}} z^{N(t)}dt\right]$, and $\mathcal{T} = E\left[\int_{\text{cycle}} 1dt\right] = \dot{\mathcal{R}}\Big|_{z=1}$. Using $(1,1)$ as our renewal state and the same arguments as in Section 5.4, we have:

$$\dot{\mathcal{R}} = \frac{z}{\lambda+\mu} + \frac{\mu}{\lambda+\mu}\left(\frac{1}{\lambda} + \frac{z}{\lambda+\alpha} + \frac{\lambda}{\lambda+\alpha}\cdot\dot{R}^L_{0,2}\right) + \frac{\lambda}{\lambda+\mu}\cdot\dot{R}^L_{1,2} \tag{19}$$

It now remains to compute the reward equations for the border states: $\dot{R}^L_{0,2}$, $\dot{R}^L_{1,2}$, and $\dot{R}^L_{2,2}$.

$$\dot{R}^L_{2,2} = \frac{z^2}{\lambda+2\mu} + \frac{\lambda}{\lambda+2\mu}\left(z\cdot\dot{R}^L_{2,2} + \dot{R}^L_{2,2}\right) \tag{20}$$

$$\dot{R}^L_{1,2} = \frac{z^2}{\lambda+\mu+\alpha} + \frac{\alpha}{\lambda+\mu+\alpha}\cdot\dot{R}^L_{2,2} + \frac{\lambda}{\lambda+\mu+\alpha}\left(z\cdot\dot{R}^L_{1,2} + \left(p^L_{1\to1}\right)\dot{R}^L_{1,2} + \left(1-p^L_{1\to1}\right)\dot{R}^L_{2,2}\right) \tag{21}$$

$$\dot{R}^L_{0,2} = \frac{z^2}{\lambda+2\alpha} + \frac{2\alpha}{\lambda+2\alpha}\cdot\dot{R}^L_{1,2} + \frac{\lambda}{\lambda+2\alpha}\left(z\cdot\dot{R}^L_{0,2} + \left(p^L_{0\to1}\right)\dot{R}^L_{1,2} + \left(1-p^L_{0\to1}\right)\dot{R}^L_{2,2}\right) \tag{22}$$

Solving the above system of linear equations and substituting $\dot{R}^L_{0,2}$ and $\dot{R}^L_{1,2}$ into Eq. (19) allows us to solve for $\widehat{N}(z)$ in closed form as follows:

$$\widehat{N}(z) = E[z^N] = \frac{\dot{\mathcal{R}}}{\mathcal{T}} = \frac{\dot{\mathcal{R}}}{\dot{\mathcal{R}}\big|_{z=1}}$$

$$= \frac{\lambda(\lambda+\alpha)(z+\lambda\dot{R}^L_{1,2}) + \mu(\alpha+\lambda(1+z+\lambda\dot{R}^L_{0,2}))}{\lambda(\lambda+\alpha)(1+\lambda T^L_{1j}) + \mu(\alpha+\lambda(2+\lambda T^L_{0j}))} \tag{23}$$

6.3 Deriving $\widetilde{\mathbf{T}}(\mathbf{s})$

For the M/M/1/setup system, we were able to derive $\widetilde{T}(s)$ directly from $\widehat{N}(z)$ via the distributional Little's Law, since the M/M/1/setup is a FIFO system. Unfortunately, the M/M/2/setup system is not FIFO, since overtaking can occur. However, we can still apply the distributional Little's Law to the queue of the M/M/2/setup since the queue *is* FIFO. The analysis of $\widehat{N_Q}(z)$ is very similar to that of $\widehat{N}(z)$ and is thus omitted:

$$\widehat{N_Q}(z) = \frac{\lambda(\lambda+\alpha)(1+\lambda\dot{R}^L_{1,2}) + \mu(\alpha+\lambda(1+z+\lambda\dot{R}^L_{0,2}))}{\lambda(\lambda+\alpha)(1+\lambda T^L_{1j}) + \mu(\alpha+\lambda(2+\lambda T^L_{0j}))} \tag{24}$$

We now apply the distributional Little's Law to get $\widetilde{T_Q}(s)$ from $\widehat{N_Q}(z)$. Finally, since $T = T_Q + S$, where $S \sim Exp(\mu)$ is the job size distribution, we have:

$$\widetilde{T}(s) = \widetilde{T_Q}(s)\cdot\frac{\mu}{s+\mu} = \widehat{N_Q}\left(1-\frac{s}{\lambda}\right)\cdot\frac{\mu}{s+\mu}$$

$$= \frac{\mu\left(\lambda(\lambda+\alpha)(1+\lambda\dot{R}^L_{1,2}) + \mu(\alpha-s+\lambda(2+\lambda\dot{R}^L_{0,2}))\right)}{(s+\mu)\left(\lambda(\lambda+\alpha)(1+\lambda T^L_{1j}) + \mu(\alpha+\lambda(2+\lambda T^L_{0j}))\right)} \tag{25}$$

6.4 Deriving $\widehat{\mathbf{P}}(\mathbf{z})$

The derivation of $\widehat{P}(z)$ is similar to that of $\widehat{N}(z)$ in Section 6.2, and is thus omitted.

$$\widehat{P}(z) = \frac{\mu(\alpha+\lambda) + \lambda(\lambda+\mu+\alpha)z^{P_{peak}}}{\mu(\alpha+\lambda) + \lambda(\lambda+\mu+\alpha) + \lambda^2(\mu T^L_{0j} + (\lambda+\alpha)T^L_{1j})} + \frac{\lambda^2(\mu T^L_{0j} + (\lambda+\alpha)T^L_{1j})z^{2P_{peak}}}{\mu(\alpha+\lambda) + \lambda(\lambda+\mu+\alpha) + \lambda^2(\mu T^L_{0j} + (\lambda+\alpha)T^L_{1j})} \tag{26}$$

7. M/M/K/SETUP

The M/M/k/setup chain shown in Fig. 1 is analyzed similarly to M/M/2/setup. The border states for M/M/k/setup are (i,k), with $0 \le i \le k$. In the M/M/k/setup, the non-repeating portion consists of $O(k^2)$ states. For $k = 1$ and $k = 2$, we did not have to explicitly write reward equations for the non-repeating states; these were implicitly folded into other equations (see, for example, the term in parentheses in Eq. (19)). However, for arbitrarily large k, it is necessary to write reward equations for the states in the non-repeating portion. We use $R^H_{i,j}$ to denote the reward earned until we reach the home state, starting from state (i,j) in the non-repeating portion. The $R^H_{i,j}$ equations will be discussed in Section 7.3.

We illustrate the RRR technique for M/M/k/setup by deriving $\widehat{N_Q}(z)$, from which we can obtain $\widetilde{T}(s)$. For a detailed demonstration of this technique for the case of $k = 3$, see [7].

One might think that analyzing the M/M/k/setup will require solving a k^{th} degree equation. This turns out to be false. Analyzing the M/M/k/setup via RRR only requires solving equations which are, at worst, quadratic.

We choose $(k-1, k-1)$ to be the renewal state. Using RRR, $\dot{\mathcal{R}}$ can be expressed as:

$$\dot{\mathcal{R}} = \frac{1 + (k-1)\mu \dot{R}^H_{k-2,k-2} + \lambda \dot{R}^L_{k-1,k}}{\lambda + (k-1)\mu} \tag{27}$$

We now derive the necessary $\mathbf{p^L_{i\to d}}$, $\dot{R}^L_{i,k}$, and $\dot{R}^H_{i,j}$ for computing $\dot{\mathcal{R}}$.

7.1 System of equations for $\mathbf{p^L_{i\to d}}$

The system of equations for $p^L_{i\to d}$ is as follows:[2]

$$p^L_{i\to i} = \frac{\lambda (p^L_{i\to i})^2 + i\mu}{\lambda + i\mu + (k-i)\alpha}, \quad (i < k) \tag{28}$$

$$p^L_{i\to d} = \frac{\lambda \left(\sum_{\ell=i}^{d} \left\{ (p^L_{i\to \ell})(p^L_{\ell\to d}) \right\} \right) + (k-i)\alpha (p^L_{i+1\to d})}{\lambda + i\mu + (k-i)\alpha}, \quad (i < d < k) \tag{29}$$

$$p^L_{i\to k} = 1 - \sum_{\ell=i}^{k-1} p^L_{i\to \ell}, \quad (i \le k) \tag{30}$$

The summation in Eqs. (29) above denotes the possible intermediate depths ℓ through which we can move from initial depth i to final depth d. The above system of equations involves linear and quadratic equations (including products of two unlike variables), and can be solved *symbolically* to find $p^L_{i\to d}$ in closed form (see Appendix B).

7.2 Deriving $\dot{\mathbf{R}}^{\mathbf{L}}_{\mathbf{i,k}}$ for the repeating portion

The system of equations for $\dot{R}^L_{i,k}$ is as follows:

$$\dot{R}^L_{0,k} = \frac{z^k + \lambda \left(z \dot{R}^L_{0,k} + \sum_{\ell=1}^{k} \left\{ (p^L_{0\to \ell})(\dot{R}^L_{\ell,k}) \right\} \right) + k\alpha \dot{R}^L_{1,k}}{\lambda + k\alpha} \tag{31}$$

$$\dot{R}^L_{i,k} = \frac{z^{k-i} + \lambda \left(z \dot{R}^L_{i,k} + \sum_{\ell=i}^{k} \left\{ (p^L_{i\to \ell})(\dot{R}^L_{\ell,k}) \right\} \right)}{\lambda + i\mu + (k-i)\alpha} + \frac{(k-i)\alpha \dot{R}^L_{i+1,k}}{\lambda + i\mu + (k-i)\alpha}, \quad (0 < i < k) \tag{32}$$

$$\dot{R}^L_{k,k} = \frac{1 + \lambda (z \dot{R}^L_{k,k} + \dot{R}^L_{k,k})}{\lambda + k\mu} \tag{33}$$

In the above, we have used the fact that $\dot{R}^L_{i,k+1} = z\dot{R}^L_{i,k}$ from Thm. 3. The above system of linear equations can be easily solved to find $\dot{R}^L_{i,k}$ in closed form (see Appendix B).

7.3 Deriving $\dot{\mathbf{R}}^{\mathbf{H}}_{\mathbf{i,j}}$ for the non-repeating portion

[2]The definition given for $p^L_{i\to d}$ applies in all cases except when $j = k$ and $d \in \{k-1, k\}$. When $j = k$, we can never end in depth k when moving one step to the left; in this case, we interpret $p^L_{i\to k}$ (or $p^L_{i\to k-1}$) as the probability that we first moved one step left *by transitioning out of a state in depth* k (or $k-1$).

The system of equations for $\dot{R}^H_{i,j}$ is as follows:

$$\dot{R}^H_{0,j} = \frac{z^j + \lambda \dot{R}^H_{0,j+1} + j\alpha \dot{R}^H_{1,j}}{\lambda + j\alpha}, \quad (j < k-1) \tag{34}$$

$$\dot{R}^H_{i,j} = \frac{z^{j-i} + \lambda \dot{R}^H_{i,j+1} + i\mu \dot{R}^H_{i,j-1} + (j-i)\alpha \dot{R}^H_{i+1,j}}{\lambda + i\mu + (j-i)\alpha}, \quad (0 < i < j < k-1) \tag{35}$$

$$\dot{R}^H_{i,i} = \frac{1 + \lambda \dot{R}^H_{i,i+1} + i\mu \dot{R}^H_{i-1,i-1}}{\lambda + i\mu}, \quad (0 < i < k-1) \tag{36}$$

$$\dot{R}^H_{i,k-1} = \frac{z^{k-1-i} + \lambda \left(\dot{R}^L_{i,k} + \sum_{\ell=i}^{k} \left\{ (p^L_{i\to \ell})(\dot{R}^H_{\ell,k-1}) \right\} \right)}{\lambda + i\mu + (k-1-i)\alpha} + \frac{i\mu \dot{R}^H_{i,k-2} + (k-1-i)\alpha \dot{R}^H_{i+1,k-1}}{\lambda + i\mu + (k-1-i)\alpha}, \quad (i < k-1) \tag{37}$$

$$\dot{R}^H_{k-1,k-1} = 0 \tag{38}$$

Eqs. (34), (35), and (36), are simply based on the rate transitions in the non-repeating portion of the Markov chain. Eqs. (37) describe the rewards earned when starting in states in the non-repeating portion of the chain that can transition to the repeating portion of the chain via the border states. When we have an arrival in one of these states, we transition to the repeating portion of the chain, and after earning some reward, return to the non-repeating portion of the chain. Finally, Eq. (38) guarantees that any transition to state $(k-1, k-1)$ will end the renewal cycle. The above system of linear equations can again be easily solved to find $\dot{R}^H_{i,j}$ in closed form (see Appendix B).

After solving for $p^L_{i\to d}$, $\dot{R}^L_{i,k}$ and $\dot{R}^H_{i,j}$, we can derive $\dot{\mathcal{R}}$, and consequently $\widehat{N_Q}(z)$, via Eq. (27). $\widetilde{T}(s)$ can then be derived by using the fact $\widetilde{T}(s) = \widetilde{T_Q}(s) \cdot \frac{\mu}{s+\mu} = \widehat{N_Q}\left(1 - \frac{s}{\lambda}\right) \cdot \frac{\mu}{s+\mu}$.

We applied the above steps to obtain a closed-form expression for $\widehat{N_Q}(z)$ for the M/M/3/setup. We refer the reader to [7] for full details.

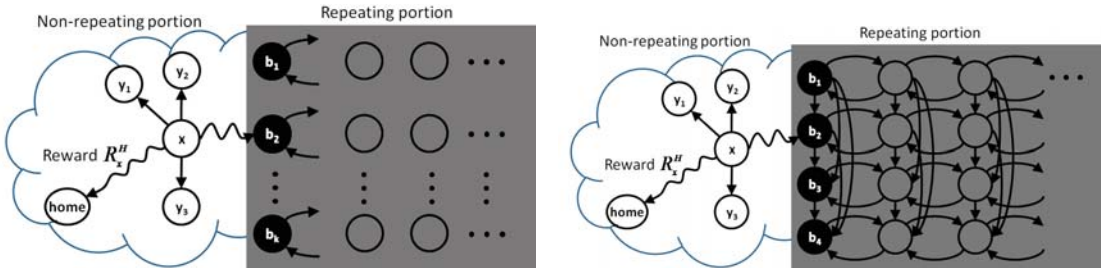
8. M/M/K/SETUP/DELAYEDOFF

The Markov chain for M/M/k/setup/delayedoff is shown in Fig. 2. Our renewal state this time will be $(k, k-1)$; thus, $\dot{\mathcal{R}}$, the reward earned when going from $(k, k-1)$ back to $(k, k-1)$ can be expressed as:

$$\dot{\mathcal{R}} = \frac{1 + (k-1)\mu \dot{R}^H_{k,k-2} + \lambda \dot{R}^L_{k,k} + \beta \dot{R}^H_{k-1,k-1}}{\lambda + (k-1)\mu + \beta} \tag{39}$$

The analysis for M/M/k/setup/delayedoff via RRR is very similar to that of M/M/k/setup in Section 7 above. In fact, since the repeating portion for the two chains is the same, the system of equations for $p^L_{i\to d}$ and $\dot{R}^L_{i,j}$ is identical, but the non-repeating portion for the two chains is different. We now set up the system of equations for solving $\dot{R}^H_{i,j}$.

8.1 Deriving $\dot{\mathbf{R}}^{\mathbf{H}}_{\mathbf{i,j}}$ for the non-repeating portion



(a) (b)

Figure 6: ***Fig. 6(a) depicts the class of Markov chains that can be analyzed via RRR. The repeating portion is highlighted in gray and the border states, b_i, are shaded black. Note that y_i are the neighbors of x. Fig. 6(b) depicts the more restrictive class of Markov chain that can be analyzed in closed-form via RRR. In this class, the horizontal transitions are skip-free and the vertical transitions are unidirectional.***

The system of equations for $\dot{R}^H_{i,j}$ is as follows:

$$\dot{R}^H_{0,j} = \frac{z^j + \lambda \dot{R}^H_{0,j+1} + j\alpha \dot{R}^H_{1,j}}{\lambda + j\alpha}, \quad (j < k-1) \tag{40}$$

$$\dot{R}^H_{i,0} = \frac{1 + \lambda \dot{R}^H_{i,1} + i\beta \dot{R}^H_{i-1,0}}{\lambda + i\beta}, \quad (0 < i \leq k) \tag{41}$$

$$\dot{R}^H_{k,j} = \frac{1 + \lambda \dot{R}^H_{k,j+1} + j\mu \dot{R}^H_{k,j-1} + (k-j)\beta \dot{R}^H_{k-1,j}}{\lambda + j\mu + (k-j)\beta}, \quad (1 \leq j < k-1) \tag{42}$$

$$\dot{R}^H_{i,j} = \frac{z^{j-i} + \lambda \dot{R}^H_{i,j+1} + i\mu \dot{R}^H_{i,j-1} + (j-i)\alpha \dot{R}^H_{i+1,j}}{\lambda + i\mu + (j-i)\alpha}, \quad (0 < i < j < k-1) \tag{43}$$

$$\dot{R}^H_{i,i} = \frac{1 + \lambda \dot{R}^H_{i,i+1} + i\mu \dot{R}^H_{i,i-1}}{\lambda + i\mu}, \quad (0 < i < k-1) \tag{44}$$

$$\dot{R}^H_{i,j} = \frac{1 + \lambda \dot{R}^H_{i,j+1} + j\mu \dot{R}^H_{i,j-1} + (i-j)\beta \dot{R}^H_{i-1,j}}{\lambda + j\mu + (i-j)\alpha}, \quad (0 < j < i < k) \tag{45}$$

$$\dot{R}^H_{i,k-1} = \frac{z^{k-1-i} + \lambda \left(\dot{R}^L_{i,k} + \sum_{\ell=i}^{k} \left\{ (p^L_{i\to\ell})(\dot{R}^H_{\ell,k-1}) \right\} \right)}{\lambda + i\mu + (k-1-i)\alpha} + \frac{i\mu \dot{R}^H_{i,k-2} + (k-1-i)\alpha \dot{R}^H_{i+1,k-1}}{\lambda + i\mu + (k-1-i)\alpha}, \quad (i \leq k-1) \tag{46}$$

$$\dot{R}^H_{k,k-1} = 0 \tag{47}$$

The above system of linear equations can again be solved to find $\dot{R}^H_{i,j}$ in closed form. This yields $\dot{\mathcal{R}}$, and consequently $\widehat{N_Q}(z)$, via Eq. (39).

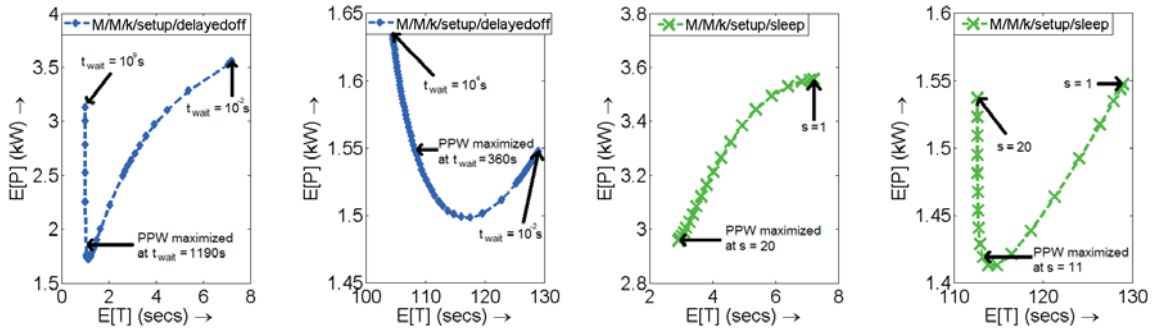
9. M/M/K/SETUP/SLEEP

The Markov chain for M/M/k/setup/sleep is shown in Fig. 3. The analysis for M/M/k/setup/sleep via RRR is again similar to that of M/M/k/setup in Section 7. The only difference is in the setup transition rate (downwards transition arrows in the Markov chain): For the M/M/k/setup, the setup rate in state (i, j) is $\alpha \cdot \min\{j - i, k - i\}$. For the M/M/k/setup/sleep, the setup rate in state (i, j) is more complicated. When $i \geq s$, the setup rate is still $\alpha \cdot \min\{j - i, k - i\}$. However, if $i < s$, the setup rate is $\omega \cdot (j - i)$ if $j \leq s$ and $\omega \cdot (s - i) + \alpha \cdot \min\{j - s, k - s\}$ if $j > s$. This can be explained based on the M/M/k/setup/sleep model description in Section 3.3 and the Markov chain in Fig. 3. Based on the above setup rates, we can easily modify the M/M/k/setup sets of equations for $p^L_{i\to d}$, $\dot{R}^L_{i,k}$ and $\dot{R}^H_{i,j}$ from Sections 7.1, 7.2 and 7.3 respectively, to represent the M/M/k/setup/sleep system of equations. The equation for $\dot{\mathcal{R}}$ will change accordingly.

10. THE GENERALIZED RECURSIVE RENEWAL REWARD TECHNIQUE

The RRR technique can be applied to a very broad class of Markov chains beyond just the M/M/k/setup. In general, RRR can reduce the analysis of any 2-dimensional, irreducible Markov chain which is repeating and infinite in one dimension (as shown in Fig. 6(a)) to the problem of solving a system of polynomial equations. Further, if in the repeating portion all horizontal transitions are skip-free and all vertical transitions are unidirectional (as shown in Fig. 6(b)), the resulting system of equations will be of degree at most two, yielding a closed-form solution. In this section we explain the application of the RRR technique to general repeating Markov chains and also provide justification for the above claims regarding Figs. 6(a) and 6(b). Throughout we will assume that the reward earned at a state, (i, j), is an affine function of i and j.

In order to apply RRR, we first partition the Markov chain into a finite non-repeating portion and an infinite repeating portion as in Fig. 6(a); in principle, this partition is not unique. Then, we fix a renewal point, or home state, within the non-repeating portion. For each state, x, in the non-repeating portion of the chain, we write an equation for the mean reward, R^H_x, earned in traveling from x to the home state. Each R^H_x is a sum of the mean reward during our residence in x and a weighted linear combination of the rewards R^H_y, where y is a neighbor of x, as in Fig. 6(a). We refer to this finite set of linear equations for the R^H_xs as (**Ia**). Since the chain is irreducible, at least one state in the non-repeating portion of the chain transitions directly to a state in the repeating portion of the chain. We refer to the states in the repeating portion that are directly accessible from the non-repeating portion as *border states*. These are shown as b_i in Fig. 6(a). We next write a set of equations for the mean reward earned in traveling from each border state to



(a) $E[S] = 1$s (b) $E[S] = 100$s (c) $E[S] = 1$s (d) $E[S] = 100$s

Figure 7: *E[P] versus E[T] for various values of t_{wait} and s.*

the home state; call this set (**Ib**). Equation sets (**Ia**) and (**Ib**) together form the linear system of equations (**I**).

Within (**Ib**), the mean reward earned when returning home from each border state b consists of two parts: (i) the mean reward earned from the time we enter b until we leave the repeating portion, and (ii) the mean reward earned from when we first exit the repeating portion until returning home. Note that (ii) is simply a weighted linear combination of R_x^Hs where the weights form the probability distribution over the set of states in the non-repeating portion that we transition to (same as the $p^L_{i\to d}$s). For (i), the reward equation can be expressed as a weighted linear combination of the rewards for the neighbors of b in the repeating portion. The fact that the chain has a repeating structure allows us to express the reward from any state in the repeating portion as a linear combination of the rewards of the border states by using "recursion theorems" (similar to Thms. 2 and 3). We also need the probability distribution over the set of states we transition to when we move left (the $p^L_{i\to d}$s). At this point, to write the equations in (**Ib**), we require solving the $p^L_{i\to d}$s. We refer to the system of equations for $p^L_{i\to d}$s as (**II**).

In writing the equations for $p^L_{i\to d}$s, we again use recursion theorems (similar to Thm. 5) that exploit the repeating structure of the Markov chain. However, this time, the equations need not be linear. This is because when moving left to depth d from depth i, we might transition through various intermediate depths. Thus, $p^L_{i\to d}$ will involve several other probability terms. Unlike rewards where we sum up intermediate terms, for probability we take a product of the intermediate terms, leading to a system of higher order polynomial equations, (**II**). Note that (**II**) does not depend on (**I**), and can be solved independently. Once we get the $p^L_{i\to d}$s by solving (**II**), we substitute these back (as constants) into the set of linear equations (**Ib**). The sets of linear equations (**Ib**) and (**Ia**) can now be jointly solved using standard techniques such as symbolic matrix inversion. This yields the mean reward earned during a renewal cycle from home to home; mean time is found analogously.

In the case of Markov chains as shown in Fig. 6(b), the probability equations, (**II**), will be of degree at most two, as in Section 7.1. This is because skip-free horizontal transitions guarantee that the probability $p^L_{i\to d}$ can be expressed as a linear sum of products of only two intermediate terms of the form $p^L_{i\to \ell} \cdot p^L_{\ell\to d}$, where ℓ represents the intermediate depths that we can transition to in going from i to d (as in Section 7.1). Further, the unidirectional vertical transitions guarantee that $i \le \ell \le d$, which ensures that the intermediate probability terms do not lead to higher-order dependencies between each other. Thus, the probabilities can be derived in closed-form by solving quadratic equations (including products of two unlike terms) in a particular "bottom-up" order as explained in Appendix B.

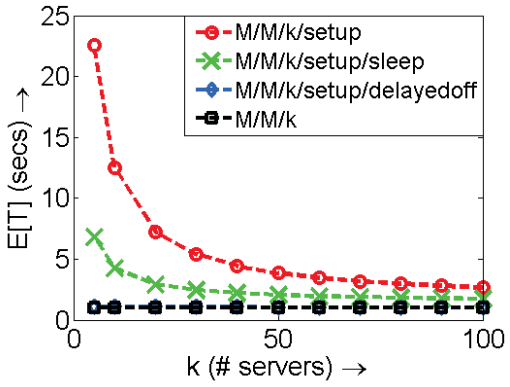
11. APPLICATIONS

In this section we use our analytical results to evaluate the performance of M/M/k, M/M/k/setup, M/M/k/setup/delayedoff and M/M/k/setup/sleep. In particular, we will be interested in the mean response time, $E[T]$, and the mean power consumption, $E[P]$, under these policies. Throughout, we assume a load of $\rho = \frac{\lambda}{k\mu} = 0.3$ (or 30% load), setup times of $\frac{1}{\alpha} = 100$s (when the server is off) and $\frac{1}{\omega} = 25$s (when the server is sleeping), and power consumption values of $P_{peak} = 200$W, $P_{idle} = 140$W, and $P_{sleep} = 14$W. These parameter values are based on empirical measurements from prior work [4, 11]. We consider job sizes with mean $E[S] = 1$s (typical web workloads [6]), $E[S] = 10$s (database queries or secured transactions), and $E[S] = 100$s (file download or upload), and system sizes ranging from $k = 5$ to $k = 100$ servers.

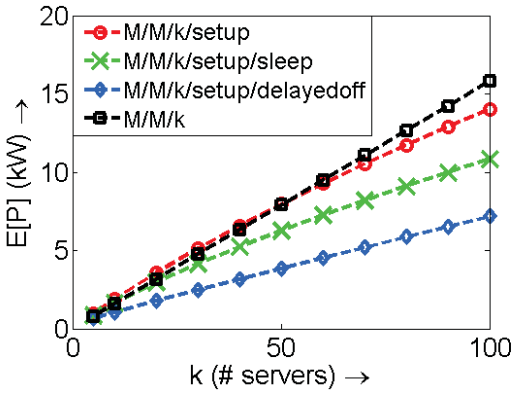
The M/M/k policy keeps k servers always on. Servers that are not busy serving jobs are left idle. The M/M/k/setup policy (see Section 3.1) immediately turns off idle servers to save power. However, restarting an off server requires a setup time of $\frac{1}{\alpha} = 100$s. The M/M/k/setup/delayedoff policy (see Section 3.2) is the same as the M/M/k/setup policy, except that idle servers wait for an exponentially distributed amount of time with mean $t_{wait} = \frac{1}{\beta}$ before turning off. The performance of this policy depends on the choice of the t_{wait} parameter. Finally, the M/M/k/setup/sleep policy (see Section 3.3) is the same as the M/M/k/setup policy, except that s of the k servers go to sleep as opposed to turning off, when idle. A sleeping server has a small setup time of $\frac{1}{\omega} = 25$s. The performance of this policy depends on the choice of the s parameter. Before comparing the above four policies, we first discuss how we choose the parameter value of t_{wait} for M/M/k/setup/delayedoff and s for M/M/k/setup/sleep.

11.1 Choosing optimal parameter values

The tradeoff between $E[P]$ and $E[T]$ for M/M/k/setup/delayedoff is shown in Figs. 7(a) and 7(b). Each plotted point represents an $(E[T], E[P])$ pair associated with a specific value of t_{wait}. Intuitively, as t_{wait} increases, $E[T]$ decreases since we avoid setup times. Moreover, before some threshold t_{wait}, $E[P]$ decreases as t_{wait} increases, because we avoid consuming power at peak rate by repeatedly putting servers

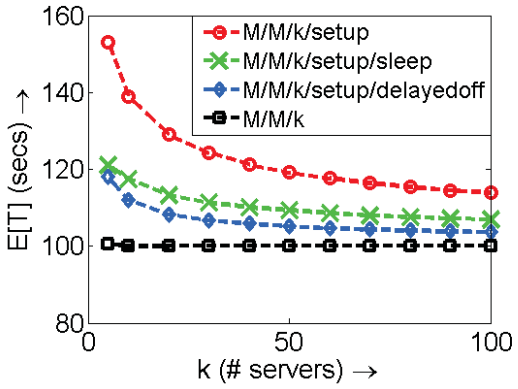


(a) E[T] versus k

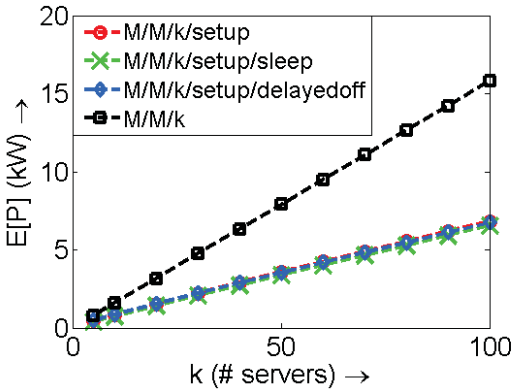


(b) E[P] versus k

Figure 8: ***Results when mean job size*** $E[S] = 1$.



(a) E[T] versus k



(b) E[P] versus k

Figure 9: ***Results when mean job size*** $E[S] = 100$.

in setup. However, beyond this threshold t_{wait}, $E[P]$ starts increasing on account of idle servers. Thus, as t_{wait} increases, we get the plots in Figs. 7(a) and 7(b), from right to left. We choose the t_{wait} value that optimizes (i.e., maximizes) the popular *Performance-Per-Watt* metric [11, 8], given by $PPW = (E[T] \cdot E[P])^{-1}$. These optimal values are shown in Figs. 7(a) and 7(b). We find that the optimal t_{wait} value decreases with an increase in $E[S]$.

Figs. 7(c) and 7(d) illustrate the tradeoff between $E[P]$ and $E[T]$ under M/M/k/setup/sleep for different values of s. Intuitively, as s increases, $E[T]$ decreases since we benefit from faster setup times afforded by sleeping servers. As s increases, $E[P]$ first decreases since we avoid the severe power penalty of longer setup times. But beyond a certain s, $E[P]$ increases on account of the sleeping servers. Thus, as s increases, we get the plots in Figs. 7(c) and 7(d), from right to left. Note that $E[P]$ monotonically decreases for the case of $E[S] = 1$s in Fig. 7(c). This is because $\frac{1}{\alpha} \gg E[S]$, and thus, the decrease in power consumption by avoiding power penalties of longer setup times outweighs the increase in power consumption because of P_{sleep}. We choose the s value that optimizes the PPW metric, as indicated in Figs. 7(c) and 7(d). We find that the optimal s value decreases with an increase in $E[S]$.

11.2 Comparison of all policies

Fig. 8 shows our results for $E[T]$ and $E[P]$ as a function of k for the case of $E[S] = 1$s. Comparing M/M/k (squares) and M/M/k/setup (circles), we see that M/M/k/setup has a much higher $E[T]$, and only a slightly lower $E[P]$. In fact, when k is low, $E[P]$ for M/M/k/setup is *higher* than that of M/M/k. This is because of the power penalty involved in the setup cost. Thus, **M/M/k/setup is not a good policy for small job sizes**. The M/M/k/setup/sleep (crosses) has lower $E[T]$ *and* lower $E[P]$ than the M/M/k/setup. Thus, **using sleep modes improves the M/M/k/setup policy**. Finally, we see that M/M/k/setup/delayedoff (diamonds) has $E[T]$ virtually as low as that of M/M/k, and has the lowest power consumption among all other policies. Thus, **M/M/k/setup/delayedoff is superior to all the other policies for small job sizes**. The reason for lower $E[P]$ under M/M/k/setup/delayedoff is because of t_{wait} which avoids unnecessary setups (and the associated power penalties).

Fig. 9 shows our results for the case of $E[S] = 100$s. The $E[T]$ results for this job size are qualitatively similar to the results for $E[S] = 1$s. The percentage difference between the $E[T]$ under different policies goes down as $E[S]$ goes up. This is because the setup time is not changing as $E[S]$ goes up, and thus, the queueing delay caused by setup times is not as severe for large $E[S]$. Note that the $E[T]$ under M/M/k/setup/delayedoff actually goes up as $E[S]$ goes up. This is a side-effect of the optimal t_{wait} setting which trades

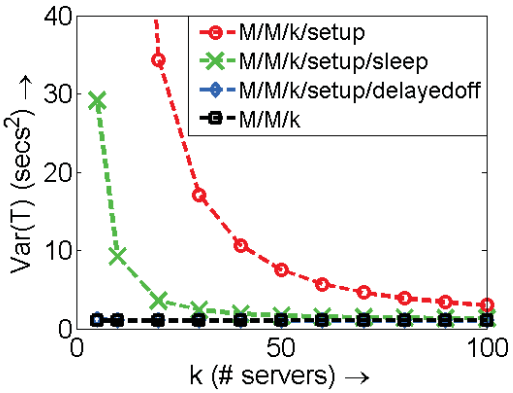


Figure 10: ***Var(T) versus k for*** $E[S] = 1s$.

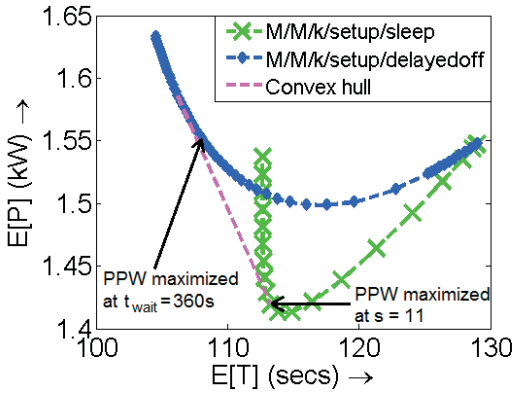


Figure 11: ***E[P] versus E[T] for various values of*** t_{wait} ***and*** s ***for mean job size*** $E[S] = 100s$.

off lower $E[P]$ at the expense of a slightly higher $E[T]$ for bigger job sizes.

The $E[P]$ results for different job sizes indicate that $E[P]$ **under M/M/k/setup and M/M/k/setup/sleep decreases with an increase in job size, and approaches the** $E[P]$ **of M/M/k/setup/delayedoff**. This is because an increase in $E[S]$ necessitates an increase in the inter-arrival time, given fixed load, ρ. Thus, servers now spend more time in the off or sleep states, and consequently, consume less power. In fact, the M/M/k/setup/sleep has *lower* $E[P]$ as compared to M/M/k/setup/delayedoff for the case of $E[S]$ = 100s. We take a closer look at these two policies in Section 11.3. Note that under M/M/k, $E[P] = k \cdot \rho \cdot P_{peak} + k \cdot (1 - \rho) \cdot P_{idle}$, which is linear in k and independent of $E[S]$.

The $E[T]$ results for these job sizes are qualitatively similar to the results for $E[S]$ = 1s. The percentage difference between the $E[T]$ under different policies goes down as $E[S]$ goes up. This is because the setup time is not changing as $E[S]$ goes up, and thus, the queueing delay caused by setup times is not as severe for large $E[S]$. Note that the $E[T]$ under M/M/k/setup/delayedoff actually goes up as $E[S]$ goes up. This is a side-effect of the optimal t_{wait} setting which trades off lower $E[P]$ at the expense of a slightly higher $E[T]$ for bigger job sizes. As mentioned in Section 7, RRR also provides closed-form solutions for higher moments of response time and power. Fig. 10 shows our results for $Var(T)$, the variability in response time, for the case of $E[S]$ = 1s. We see that $Var(T)$ follows the same trends as $E[T]$ in Fig. 8(a). Note that $Var(T)$ is close to 1 for M/M/k and M/M/k/setup/delayed-off. Also, $Var(T)$ converges to 1 for all policies for high k. This is because $Var(T)$ converges to $Var(S)$ (no queueing delay) in these cases, and since S is exponentially distributed with mean $E[S]$ = 1s, we get $Var(T) \to Var(S) = 1s^2$.

All the results above assumed exponential setup times and exponential delay times. However, in real-world scenarios, these times would be deterministic. We use simulations to find $E[T]$ and $E[P]$ under deterministic setup times for all the above cases. We find that the relative ordering of the policies and the trends in $E[T]$ and $E[P]$ do not change significantly, despite the fact that all values become slightly higher due to the setup rates no longer being additive.

11.3 A closer look at M/M/k/setup/delayedoff versus M/M/k/setup/sleep

Fig. 11 shows the tradeoff between $E[P]$ and $E[T]$ for M/M/k/setup/delayedoff and M/M/k/setup/sleep for $E[S]$ = 100s. These plots are identical to Figs. 7(b) and 7(d). We see that no policy dominates the other. If we are more concerned about reducing $E[P]$, M/M/k/setup/sleep is the better choice. However, if we are more concerned about reducing $E[T]$, M/M/k/setup/delayedoff is the better choice. Interestingly, by taking a probabilistic mixture of the two policies, we can find additional policies that are superior to the M/M/k/setup/delayedoff and the M/M/k/setup/sleep. The probabilistic mixture can be obtained by taking the convex hull of the two policies, as shown by the dashed line in Fig. 11. This suggests the potential for a policy that combines M/M/k/setup/sleep with delayedoff.

12. CONCLUSION

In this paper we develop a new analysis technique, Recursive Renewal Reward (RRR), which allows us to solve the M/M/k/setup class of Markov chains. RRR is very intuitive, easy to apply, and can be used to analyze many important Markov chains that have a repeating structure. RRR combines renewal reward theory with the development of recursion theorems for the Markov chain to yield exact, closed form results for metrics of interest such as the transform of time in system and the transform of power consumed by the system. RRR reduces the solution of the M/M/k/setup chains to solving k quadratic equations and a system of $O(k^2)$ linear equations. On an Intel Core i5-based processor machine we found RRR to be almost 5-10 times faster than the iterative matrix-analytic based methods, when using standard MATLAB implementations of both methods.

While we have only considered the M/M/k/setup, the M/M/k/setup/delayedoff, and the M/M/k/setup/sleep in this paper, we have also been able to use RRR for the derivation of exact, closed-form solutions for other important Markov chains with a repeating structure such as: (i) M/M/k/stag [10], wherein at most one server can be in setup, (ii) M/M/k/setup-threshold, wherein the servers are turned on and off based on some threshold for number of jobs in queue, (iii) M/M/k/disasters, wherein the system can empty abruptly due to disasters, and (iv) M/E_2/k, where the job size distribution is Erlang-2. We have also been able to apply RRR to analyze other Markov chains such as: (i) M$_t$/M/1, where the arrival process is Poisson with a time dependent parameter, (ii) M/H_2/k, where the job size dis-

tribution is a 2-phase hyperexponential, and (iii) $M^x/M/k$, where there is a Poisson batch arrival process. In the above three cases, RRR reduces the analysis to solving a system of polynomial equations with degree > 2. In general, RRR should be able to reduce the analysis of any 2-dimensional Markov chain (with an affine reward function), which is finite in one dimension and infinite (with repeating structure) in the other, to solving a system of polynomial equations.

While not shown in this paper, it is possible to derive an explicit rate matrix for the M/M/k/setup, which leads to closed-form expressions for the limiting probabilities. While RRR does not utilize the rate matrix in any way, we believe that the set of Markov chains that can be solved in closed form via RRR should have an explicit rate matrix.

13. REFERENCES

[1] I. Adan and J. Resing. A class of Markov processes on a semi-infinite strip. Technical Report 99-03, Eindhoven University of Technology, Department of Mathematics and Computing Sciences, 1999.

[2] I. Adan and J. van der Wal. Combining make to order and make to stock. *OR Spektrum*, 20:73–81, 1998.

[3] J. R. Artalejo, A. Economou, and M. J. Lopez-Herrero. Analysis of a multiserver queue with setup times. *Queueing Syst. Theory Appl.*, 51(1-2):53–76, 2005.

[4] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12):33–37, 2007.

[5] M. Castellanos, F. Casati, M.-C. Shan, and U. Dayal. iBOM: A platform for intelligent business operation management. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 1084–1095, Tokyo, Japan, 2005.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *Proceedings of twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 205–220, Stevenson, WA, 2007.

[7] A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf. Exact Analysis of the M/M/k/setup Class of Markov Chains via Recursive Renewal Reward. Technical Report CMU-CS-13-105, Carnegie Mellon University, 2013.

[8] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. Kozuch. Optimality Analysis of Energy-Performance Trade-off for Server Farm Management. *Performance Evaluation*, 67:1155–1171, 2010.

[9] A. Gandhi and M. Harchol-Balter. How Data Center Size Impacts the Effectiveness of Dynamic Power Management. *49th Annual Allerton Conference on Communication, Control, and Computing*, 2011.

[10] A. Gandhi, M. Harchol-Balter, and I. Adan. Server farms with setup costs. *Performance Evaluation*, 67:1123–1138, 2010.

[11] A. Gandhi, M. Harchol-Balter, and M. Kozuch. Are Sleep States Effective in Data Centers? *3rd IEEE International Green Computing Conference*, 2012.

[12] T. Horvath and K. Skadron. Multi-mode energy management for multi-tier server clusters. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, pages 270–279, Toronto, Ontario, Canada, 2008.

[13] J. Keilson and L. Servi. A distributional form of little's law. *Operations Research Letters*, 7(5):223 – 227, 1988.

[14] J. Kim and T. S. Rosing. Power-aware resource management techniques for low-power embedded systems. In S. H. Son, I. Lee, and J. Y.-T. Leung, editors, *Handbook of Real-Time and Embedded Systems*. Taylor-Francis Group LLC, 2006.

[15] L. Kleinrock. *Queueing Systems, Volume I: Theory*. Wiley-Interscience, 1975.

[16] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz. NapSAC: Design and implementation of a power-proportional web cluster. In *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, Green Networking '10, pages 15–22, New Delhi, India, 2010.

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

[18] Y. Levy and U. Yechiali. An M/M/s queue with servers' vacations. *INFOR*, 14:153–163, 1976.

[19] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating server idle power. In *Proceeding of the 14th international conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '09, pages 205–216, Washington, DC, 2009.

[20] I. Mitrani. Managing performance and power consumption in a server farm. *Annals of Operations Research*, pages 1–14, 2012.

[21] W. Qin and Q. Wang. Modeling and control design for performance management of web servers via an IPV approach. *IEEE Transactions on Control Systems Technology*, 15(2):259–275, March 2007.

[22] A. Riska and E. Smirni. M/G/1-type Markov processes: A tutorial. In *Performance Evaluation of Complex Systems: Techniques and Tools*, pages 36–63. Springer, 2002.

[23] N. Tian, Q.-L. Li, and J. Gao. Conditional stochastic decompositions in the M/M/c queue with server vacations. *Stochastic Models*, 15(2):367–377, 1999.

[24] B. Van Houdt and J. van Leeuwaarden. Triangular M/G/1-Type and Tree-Like Quasi-Birth-Death Markov Chains. *INFORMS Journal on Computing*, 23(1):165–171, 2011.

[25] J. Van Leeuwaarden and E. Winands. Quasi-birth-and-death processes with an explicit rate matrix. *Stochastic models*, 22(1):77–98, 2006.

[26] P. Welch. On a generalized $M/G/1$ queueing process in which the first customer of each busy period receives exceptional service. *Operations Research*, 12:736–752, 1964.

[27] X. Xu and N. Tian. The M/M/c Queue with (e, d) Setup Time. *Journal of Systems Science and Complexity*, 21:446–455, 2008.

[28] Z. G. Zhang and N. Tian. Analysis on queueing systems with synchronous vacations of partial servers. *Performance Evaluation*, 52(4):269 – 282, 2003.

APPENDIX
A. RECURSION THEOREMS

THEOREM 1 (RECURSION THEOREM FOR MEAN TIME)
For the M/M/k/setup, the mean time to move one step left from state (i,j), $T^L_{i,j}$, is the same for all $j \geq k$.

PROOF. For any $j \geq k$, observe that when moving one step left from any state (i,j), we only visit states with level j or greater, until the final transition to level $j-1$. Hence, $T^L_{i,j}$ depends only on the structure of the "subchain" of the M/M/k/setup consisting of levels $\{j, j+1, \ldots\}$, including transition rates to level $j-1$. Now consider the subchain for each $j \geq k$; these subchains are isomorphic, by the fact that the chain is repeating from level k onward. Hence, the time to move one step left is the same regardless of the initial level $j \geq k$. □

THEOREM 2 (RECURSION THEOREM FOR MEAN REWARD)
For the M/M/k/setup, the mean reward earned in moving one step left from state $(i, j+1)$, $R^L_{i,j+1}$, satisfies $R^L_{i,j+1} = R^L_{i,j} + T^L_{i,j}$ for all $j \geq k$, where the reward tracks the number of jobs in the system.

PROOF. Consider the process of moving one step left from a given state (i,j) where $j \geq k$. At the same time, consider the same process where everything is shifted over one level to the right, so that the initial state is $(i, j+1)$ At any point in time, the number of jobs seen by the second process is exactly one greater than that seen by the first process. Therefore, the total number of jobs accumulated (total reward) during the second process is $T^L_{i,j}$ greater than that of the first process, since the duration of both processes is $T^L_{i,j}$ by Theorem 1. □

THEOREM 3 (RECURSION THEOREM FOR TRANSFORM OF REWARD)
For the M/M/k/setup, $\dot{R}^L_{i,j+1} = z \cdot \dot{R}^L_{i,j}$, for all $j \geq k$, where $\dot{R}$ tracks the z-transform of the number of jobs in the system.

PROOF. The proof is identical to that of Theorem 2, except that in any moment in time the second process (starting in level $(i, j+1)$) earns z times as much reward as the first process (starting at (i,j)). □

THEOREM 4 (RECURSION THEOREM FOR TRANSFORM OF POWER)
For the M/M/k/setup, $\dot{E}^L_{i,j+1} = \dot{E}^L_{i,j} = T^L_{i,j} \cdot z^{k \cdot P_{peak}}$, for all $j \geq k$.

PROOF. When $j \geq k$, all k servers are either on or in setup, putting power consumption at $k \cdot P_{peak}$. So the transform of power usage is $z^{k \cdot P_{peak}}$, yielding $\dot{E}^L_{i,j} = T^L_{i,j} \cdot z^{k \cdot P_{peak}}$. It then follows immediately from Theorem 1 that $\dot{E}^L_{i,j+1} = \dot{E}^L_{i,j}$. □

THEOREM 5 (RECURSION THEOREM FOR PROBABILITY)
For the M/M/k/setup, for each $0 \leq d \leq k$ and for each $0 \leq i \leq k$, $p^L_{i \to d}$ is the same for all $j \geq k$.

PROOF. Recall that $p^L_{i \to d}$ is the probability that, given that we start at depth i, we end at depth d when moving one step to the left, except when $j = k$ and $d \in \{k-1, k\}$; in these cases we interpret $p^L_{i \to k}$ (or $p^L_{i \to k-1}$) as the probabilities that we first moved one step left by *transitioning out of a state* in depth k (or $k-1$).

As with $T^L_{i,j}$, $p^L_{i \to d}$ depends only on the structure of the "subchain" consisting of levels $\{j, j+1, \ldots\}$, including transition rates to level $j-1$. Since for all $j \geq k$ the resulting subchains are isomorphic, $p^L_{i \to d}$ must be the same for all $j \geq k$. □

B. SOLUTION OF THE SYSTEM OF EQUATIONS FOR M/M/K/SETUP

The steps below illustrate how to solve the system of equations for M/M/k/setup. All of the operations in the steps below can be performed *symbolically* to obtain closed-form results.

B.1 Solving for $\mathbf{p^L_{i \to d}}$

The system of equations for $p^L_{i \to d}$ consists of equation sets (28), (29) and (30). Eqs. (28) are k quadratic equations, each in one variable: $p^L_{0 \to 0}, p^L_{1 \to 1}, \ldots, p^L_{k-2 \to k-2}, p^L_{k-1 \to k-1}$. Thus, we can solve each equation easily using the quadratic formula. It can be easily shown that among the two roots of each equation, the greater root exceeds 1, and is thus disregarded. The lesser root can be shown to lie in the interval $[0, 1)$, making it the unique solution of interest to the quadratic equation. Note that $p^L_{0 \to 0} = 0$, as expected (we cannot move to the left when we have no servers on).

The set of equations (29) is a collection of $O(k^2)$ equations involving linear terms and products of two unlike variables. However, the structure of this system of equations reduces solving the system to solving a set of linear equations via back substitution. Consider solving this set of equations for the unknown values of $p^L_{i \to d}$ in this order:

$$p^L_{k-1 \to k-1}, p^L_{k-2 \to k-2}, p^L_{k-2 \to k-1}, \ldots, p^L_{0 \to 1}\, p^L_{0 \to 2}, \ldots, p^L_{0 \to k-1}$$

That is, solving from greatest $(k-1)$ to least (0) "original depth," but within each original depth, solving from least to greatest "target depth." Solving in this order, each equation we solve will only have one unknown, as all other variables will already have been solved for in an earlier step (including the $p^L_{i \to i}$ from Equations (28)), so these variables can be viewed as coefficients and constant terms. Once we have solved Equations (29), we can easily solve Equations (30), yielding $p^L_{0 \to k}, p^L_{1 \to k}, \ldots, p^L_{k-1 \to k}$, by taking complements. It follows that all $p^L_{i \to d}$ can be solved in closed forms that are, at worst, linear combinations of radicals (i.e., square roots).

B.2 Solving for $\dot{\mathbf{R}}^{\mathbf{L}}_{\mathbf{i,k}}$

The system of equations for $\dot{R}^L_{i,k}$ consists of Equations (31) and (32), and Eq. (33). This system is a collection of $(k+1)$ linear equations with $(k+1)$ unknowns. Although we could solve this system using standard linear algebraic techniques, the structure of this system suggests an even simpler approach using back substitution. Solving for each $\dot{R}^L_{i,k}$ only requires knowing the $\dot{R}^L_{i,\ell}$ such that $\ell \in \{i+1, \ldots, k\}$. Eq. (33) readily gives us $\dot{R}^L_{k,k}$. Thus, we can now solve for $\dot{R}^L_{k-1,k}$, then $\dot{R}^L_{k-2,k}$, and so on. In this way, each $\dot{R}^L_{i,k}$ is found by solving a linear equation for one unknown variable.

B.3 Solving for $\dot{\mathbf{R}}^{\mathbf{H}}_{\mathbf{i,j}}$

The system of equations for $\dot{R}^H_{i,j}$ consists of Equations (34), (35), (36) and (37), and Eq. (38). This system is a collection of $O(k^2)$ dependent linear equations with just as many unknowns. Unlike the earlier systems of equations, there is no apparent structure we can exploit, so the system can be solved via standard linear algebraic techniques such as (symbolic) matrix inversion.