Bounding of Performance Measures for Threshold-Based Queuing Systems: Theory and Application to Dynamic Resource Management in Video-on-Demand Servers

Leana Golubchik, *Member*, *IEEE Computer Society*, and
John C.S. Lui, *Member*, *IEEE Computer Society*

Abstract—In this paper, we consider a *K*-server threshold-based queuing system with hysteresis in which the number of active servers is governed by a *forward threshold* vector $F = (F_1, F_2, \ldots, F_{K-1})$ (where $F_1 < F_2 < \cdots < F_{K-1}$) and a *reverse threshold* vector $R = (R_1, R_2, \ldots, R_{K-1})$ (where $R_1 < R_2 < \cdots < R_{K-1}$). There are many applications where a threshold-based queuing system can be of great use. The main motivation for using a threshold-based approach in such applications is that they incur significant server setup, usage, and removal costs. As in most practical situations, an important concern is not only the system performance, but rather its cost/performance ratio. The motivation for use of hysteresis is to control the cost during momentary fluctuations in workload. An important and distinguishing characteristic of our work is that, in our model, we consider the *time to add a server to be nonnegligible*. This is a more accurate model, for many applications, than previously considered in other works. Our goal in this work is to develop an efficient method for computing the steady state probabilities of a multiserver threshold-based queuing system with hysteresis, which will, in turn, allow computation of various performance measures. We also illustrate how to apply this methodology in evaluation of the performance of a Video-on-Demand (VOD) storage server which dynamically manages its I/O resources.

Index Terms—Threshold-based systems, Markov chains, error bounds, matrix-geometric, video-on-demand systems.

1 Introduction

In this paper, we consider a *K*-server threshold-based queuing system with hysteresis in which the number of servers, employed for serving customers, is governed by a *forward threshold* vector $\boldsymbol{F} = (F_1, F_2, \ldots, F_{K-1})$ (where $F_1 < F_2 < \cdots < F_{K-1}$) and a *reverse threshold* vector $\boldsymbol{R} = (R_1, R_2, \ldots, R_{K-1})$ (where $R_1 < R_2 < \cdots < R_{K-1}$). This multiserver queuing system behaves as follows: A customer arriving to an empty system is served by a single server. A new arrival to a system with F_i customers already there forces a *noninstantaneous* activation of one additional server. A departure from a system which leaves R_i customers behind forces a removal of one server.

The main motivation for using a threshold-based approach is that many systems incur significant server setup, usage, and removal costs. As in most practical situations, an important concern of a system designer is not only the system performance but rather its cost/performance ratio. More specifically, under light loads, it is not desirable to unnecessarily operate many servers due to the incurred setup and usage costs. On the other hand, it is also not desirable for a system to exhibit very long delays, which can result due to lack of servers under heavy loads. One approach to improving the cost/performance ratio of a system is to react to changes in workload through the use of thresholds. For instance, one can maintain the expected job response time in a system at an acceptable level and, at the same time, maintain an acceptable cost for operating that system by dynamically adding or removing servers, depending on the system load.

There are many applications where a threshold-based queuing system can be of great use, e.g., in transport protocols of communication networks [15], where several transport-layer connections are multiplexed onto a single network layer connection. Whenever the traffic exceeds a certain threshold in the network-layer connection, another network-layer connection can be created to serve the incoming traffic from the transport layer. Using such a control mechanism, severe degradations in throughput and delay can be avoided. At the same time, operational costs can be kept at an acceptable level. Another example application is a system providing information query service via the Internet. As the number of queries increases, the number of servers needed to maintain certain (acceptable) system response time characteristics is also increased. Since the cost of setting up server connections can be significant,[1] the use of a threshold-based approach can result in a cost-controlled creation and deletion of these connections

- L. Golubchik is with the Department of Computer Science and UMIACS, University of Maryland at College Park, College Park, MD 20742. E-mail: leana@cs.umd.edu.
- J.C.S. Lui is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.

Manuscript received 23 Apr. 1999; accepted 2 Aug. 2001.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 109686.

1. For instance, it may be necessary to broadcast information about the newly added server to the already active servers in the system.

according to the changes in the workload. Finally, another application is a Video-On-Demand (VOD) system, where one approach to addressing skews in data access is replication of popular objects [34]. In order to have better load balancing characteristics in the system, the number of replicas of each object should change as the object access patterns change. Since the cost of altering the number of replicas is significant, the use of a threshold-based approach can result in a cost-controlled creation and deletion of these replicas according to the changes in the access patterns. Thus, the model presented in this paper and its solution will be beneficial for many systems and applications which manage their resources dynamically in order to control the cost/performance ratio. In this work, we use the VOD system as an example (see Section 7 for details).

As in the case of electronic circuits that are prone to oscillation effects, a "simple" threshold-based system may not suffice. In a computer system, one reason for avoiding oscillations are the above-mentioned server setup and removal costs, i.e., oscillations coupled with nonnegligible server setup and removal costs can result in a poor cost/performance ratio of a system. More specifically, it is desirable to add servers only when a system is moving toward a heavily loaded operation region and it is desirable to remove servers only when a system is moving toward a lightly loaded operation region. It is not desirable to alter the number of servers during "momentary" changes in the workload. Such oscillation regions can be avoided (as in electronic circuits) by adding a hysteresis to the system—hence, the motivation for looking for efficient analysis techniques of threshold-based queuing systems with hysteresis.

A threshold-based queuing system with hysteresis is defined by the forward and reverse threshold vectors (see Section 2 for details). The actual values, or, rather, what are "good" values for these vectors, are a function of many factors, such as the values of server setup, usage, and removal costs as well as the characteristics of the arrival process and the service rates. Although the question of optimal or "good" values for the threshold vectors is an very interesting one, it is outside the scope of this paper and is the topic of future work. However, our solution method, due to its intuitive nature, should facilitate accessible experimentation techniques for investigating the "goodness" of various threshold values.

In this paper, we present an efficient technique for computing tight performance bounds for a Markov chain model of a threshold-based queuing system with hysteresis as well as illustrate its application to performance evaluation of a Video-on-Demand system. We begin with a very brief survey of the published literature on the threshold-based queuing problem. A two-server system is considered in [17], [20], and [27]. An approximate solution for solving a degenerate form of this problem (where all thresholds are set to zero) is presented in [10], [12]; an approximate solution for a system that employs (nonzero) thresholds is presented in [29] (but without hysteresis). In [11], the authors solve a limited form of the multiserver threshold-based queuing system with hysteresis using the Green's function method [8], [13], [14]. They give a closed-form solution for a K-server system, when the servers are homogeneous, and, for a 2-server system, when the servers are heterogeneous. The authors experience difficulties in extending the Green's function method beyond two heterogeneous servers. In [23] we give exact solutions, using stochastic complementation [26], for the K-server homogeneous, heterogeneous, and bulk arrival variations of the multiserver threshold-based queuing system with hysteresis—no restrictions are placed on the number of servers or the bulk sizes or the size of the waiting room. We believe that stochastic complementation is a more intuitive and a more easily extensible method and it is exploited in this work as well.

Specifically, we consider and solve a homogeneous multiserver threshold-based queuing system with hysteresis. *We place no restrictions on the size of the waiting room or on the number of servers*. The contributions of this work are as follows: To the best of our knowledge, *none* of the works described above consider the *time to activate a server*. Since, for many applications, this time is nonnegligible, we consider it an important and distinguishing characteristic of our work. We first introduced the noninstantaneous server activation model in [6]. This model has an exact solution for computing the steady state probabilities using the matrix geometric method [30]. However, the exact solution of this problem, obtained using the matrix-geometric approach, is computationally prohibitive for moderate to large numbers of servers. Thus, the main contribution of our work is an *efficient* solution of a threshold-based queuing system with hysteresis obtained through a computation of *tight* and *provable* performance *bounds*. More specifically, we compute the steady state probabilities of the bounding models using a combination of stochastic complementation [26] and the matrix geometric [30] methods. Given the steady state probabilities, we can compute tight bounds on various performance measures of interest. The ease with which we are able to obtain these bounds demonstrates the extensibility of our method. In this paper, we extend these results in [6] by: 1) deriving a proof for the fact that these models indeed results in bounds on several performance measures of interest, 2) deriving alternative approaches to constructing bounding models, and 3) illustrate how to apply our methodology to a dynamic resource management problem for Video-on-Demand systems.

The remainder of this paper is organized as follows: In Section 2, we give a detailed description of our model. In Section 3, we present background information which is useful in solving the model of Section 2. Our general approach to solving the model is given in Section 4. Due to the complexity of the solution, it is desirable to consider more cost efficient approaches to obtaining performance measures. This is done through bounding techniques, which are presented in Sections 5 and 6. In Section 7, we apply our modeling technique to evaluating the performance of a Video-on-Demand (VOD) storage server. The goodness of the bounds and numerical results are discussed in Section 8, where we use the VOD system as an example application. Finally, our conclusions are given in Section 9.

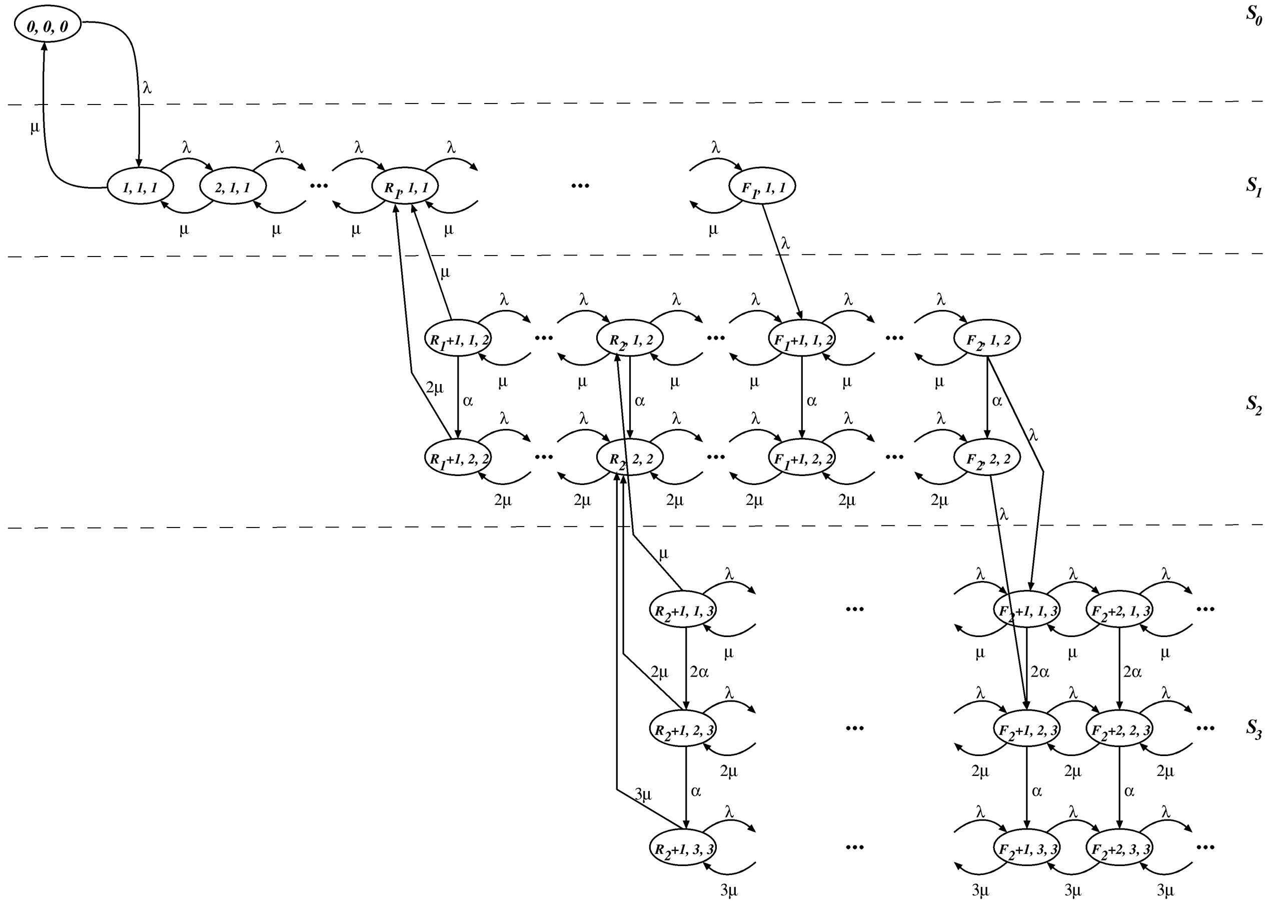


Fig. 1. State transition diagram for $K = 3$.

2 Model

In this section, we describe our model, which is illustrated in Fig. 1. Specifically, we consider a multiserver threshold-based queuing system with hysteresis that can be defined as follows: There are K homogeneous servers in the system, where K is unrestricted, each with an exponential service rate μ. The customer arrival process is Poisson with rate λ. Addition and removal of servers in this queuing system is governed by the forward and the reserve threshold vectors $\boldsymbol{F} = (F_1, F_2, \ldots, F_{K-1})$ and $\boldsymbol{R} = (R_1, R_2, \ldots, R_{K-1})$, where $F_1 < F_2 < \cdots < F_{K-1}$, and $R_1 < R_2 < \cdots < R_{K-1}$, and $R_i < F_i$ for all i. There are multiple ways to create a total order between the F_is and the R_is; for clarity and ease of presentation, in the remainder of this paper (unless otherwise stated), we assume that $R_{i+1} < F_i \forall i$. However, our solution technique can be easily extended to all other cases as well. Note that, unlike in [11], [23], the addition of a server is not instantaneous, but is governed by a Poisson process with a rate α (refer to Fig. 1). This is motivated by the fact that in many applications addition of a new server takes a nonnegligible amount of time (for instance, as in the case of the VOD application discussed in Section 7). The use of a threshold-based approach can result in a cost-controlled addition and removal of servers.

Given a K-server threshold-based queuing system with hysteresis, we can construct a corresponding Markov process $\mathcal{M}$ with the following state space $\mathcal{S}$:

$$\mathcal{S} = \{(N, N_s, L) \mid N \geq 0,\ N_s \in \{0, 1, 2, \ldots, K\}, L \in \{0, 1, 2, \ldots, K\}\},$$

where N is the number of customers in the queuing system, N_s is the number of activated servers, and L is the level to which the state belongs. Specifically, all states at level L correspond to the state of the system where, according to the threshold vectors, L servers "should be" active but may not be since server activation is not instantaneous in our model. (The "level" part of the state description is somewhat artificial at this point, but will become useful later in constructing a solution to this model.) We also use the convention that an empty system is in state $(0, 0, 0)$. Fig. 1 illustrates the state transition diagram where $K = 3$ (the S_i notation in the figure will be defined in Section 4). Formally, the transition structure of $\mathcal{M}$ with K homogeneous servers, where K is unrestricted, can be specified as follows:

$$\begin{array}{lll}
(0,0,0) & \longrightarrow & (1,1,1) \\
\lambda & & \\
(i,j,l) & \longrightarrow & (i+1,j,l+1) \\
\lambda \mathbf{1}\{(i = F_k \in \boldsymbol{F}) \wedge (l = k)\} & & \\
(i,j,l) & \longrightarrow & (i+1,j,l) \\
\lambda \mathbf{1}\{\ (i \notin \boldsymbol{F}) \vee (i = F_k \in \boldsymbol{F}) & & \\
\wedge (l \neq k)\} & & \\
(i,j,l) & \longrightarrow & (i,j+1,l) \\
(l-j)\alpha \mathbf{1}\{\ (l-j) > 0\ \} & & \\
(i,j,l) & \longrightarrow & \\
(i-1, \min(j, l-1), l-1) & & \\
j\mu \mathbf{1}\{\ (i-1 = R_k \in \boldsymbol{R}) & & \\
\wedge (l = k+1)\ \} & & \\
(i,j,l) & \longrightarrow & (i-1,j,l) \\
j\mu \mathbf{1}\{(i \geq 1 \wedge (i,j,l) \neq (1,1,1)) & & \\
\wedge(\ (i-1 \notin \boldsymbol{R}) \vee & & \\
(i-1 = R_k \in \boldsymbol{R}) & & \\
\wedge (l \neq k+1)\ )\} & & \\
(1,1,1) & \longrightarrow & (0,0,0) \\
\mu, & &
\end{array} \qquad (1)$$

where $\mathbf{1}\{x\}$ is an indicator function of x and the last column above indicates the rate at which the corresponding transition occurs.

3 Background

In this section, we briefly describe background information used in the remainder of the paper. In Section 3.1, we describe the matrix geometric approach, which is the technique we use to compute an exact solution for the model of Section 2. In Section 3.2, we describe the stochastic complement approach which is used in solving the upper and lower bound models.

3.1 Matrix Geometric Approach

A Markov process, $\mathcal{G}$, has a quasi birth-death version of the matrix geometric form [30] if the state space of $\mathcal{G}$ can be partitioned into disjoint sets B_i, $i \in \{0, 1, \cdots\}$, such that the generator matrix of $\mathcal{G}$ has the following form:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{B}_{1,0} & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad (2)$$

where $\mathbf{B}_{0,0}$ represents transition rates for states in B_0, $\mathbf{B}_{0,1}$ represents transition rates from states in B_0 to states in B_1, $\mathbf{B}_{1,0}$ represents transition rates from states in B_1 to states in B_0, $\mathbf{A}_1$ represents transition rates for states in B_i (where $i \geq 1$), $\mathbf{A}_0$ represents transition rates from states in B_i to states in B_{i+1} (where $i \geq 1$), and $\mathbf{A}_2$ represents transition rates from states in B_i to states in B_{i-1} (where $i \geq 2$). The solution of this system can be obtained by solving the following matrix equation:

$$\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2 = \mathbf{0},$$

where the matrix $\mathbf{R}$ can be computed using the following iterative procedure:

$$\mathbf{R}(0) = \mathbf{0} \qquad (3)$$

$$\mathbf{R}(n+1) = -\mathbf{A}_0\mathbf{A}_1^{-1} - \mathbf{R}^2(n)\mathbf{A}_2\mathbf{A}_1^{-1} \quad n = 0, 1, \ldots. \qquad (4)$$

Let π_i be the steady state probability vector for states in the set B_i, where $i \geq 0$. Then,

$$\pi_i = \pi_1 \mathbf{R}^{i-1} \qquad i = 2, 3, \ldots. \qquad (5)$$

For the states in B_0 and B_1, we have the following relationship:

$$\begin{aligned} \pi_0 \mathbf{B}_{0,0} + \pi_1 \mathbf{B}_{1,0} &= \mathbf{0} \\ \pi_0 \mathbf{B}_{0,1} + \pi_1 \mathbf{A}_1 + \pi_2 \mathbf{A}_2 &= \mathbf{0}, \end{aligned} \qquad (6)$$

which can be written in matrix form as

$$(\pi_0, \pi_1) \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} \\ \mathbf{B}_{1,0} & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 \end{bmatrix} = \mathbf{0}, \qquad (7)$$

where we substitute $\pi_2 = \pi_1 \mathbf{R}$ in (5) to obtain the submatrix in the lower righthand corner of (7). To determine the steady state probabilities of all states, we need the following normalization constraint:

$$1 = \pi_0 e + \pi_1 \sum_{j=1}^{\infty} \mathbf{R}^{j-1} e = \pi_0 e + \pi_1 (\boldsymbol{I} - \mathbf{R})^{-1} e,$$

where e is the column vector with all entries equal to 1. We can then determine π_i by solving the following system of linear equations:

$$(\pi_0, \pi_1) \begin{bmatrix} e & \mathbf{B}^*_{0,0} & \mathbf{B}_{0,1} \\ (\boldsymbol{I} - \mathbf{R})^{-1} e & \mathbf{B}^*_{1,0} & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 \end{bmatrix} = [1, \mathbf{0}], \qquad (8)$$

where $\mathbf{M}^*$ is the matrix $\mathbf{M}$ but with its first column removed.

3.2 Stochastic Complementation

In this section, we briefly review the concept of stochastic complementation [26], [28], [24]. For the purposes of this presentation, we assume a discrete state space, discrete time, ergodic Markov chain (throughout the paper we will also consider continuous time Markov processes; however, there is a simple transformation between the two via uniformization [7]). Given an irreducible discrete time Markov chain, $\mathcal{M}$, with state space S, let us partition this state space into two disjoint sets A and B. Then, the one-step transition probability matrix of $\mathcal{M}$ is

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_{A,A} & \boldsymbol{P}_{A,B} \\ \boldsymbol{P}_{B,A} & \boldsymbol{P}_{B,B} \end{bmatrix}$$

and $\pi = [\pi_A, \pi_B]$ is the corresponding steady state probability vector of $\mathcal{M}$. In what follows, we define the notion for stochastic complementation and quote some useful results [26].

Definition 1. *The stochastic complement of $\boldsymbol{P}_{A,A}$, denoted by $\boldsymbol{C}_{A,A}$, is*

$$\boldsymbol{C}_{A,A} = \boldsymbol{P}_{A,A} + \boldsymbol{P}_{A,B}[\boldsymbol{I} - \boldsymbol{P}_{B,B}]^{-1}\boldsymbol{P}_{B,A}. \qquad (9)$$

Theorem 1. *The stochastic complement is always a stochastic matrix and the associated Markov chain is always irreducible, if the original Markov chain is irreducible.*

Theorem 2. *Let $\pi_{|A}$ be the stationary state probability vector for the stochastic complement $C_{A,A}$, then*

$$\pi_{|A} = 1/(\pi_A e)\pi_A. \tag{10}$$

The above theorems imply that the stationary state probabilities of the stochastic complement are the *conditional state probabilities* of the associated states of the original Markov chain.

Let $\mathrm{diag}(v)$ be a diagonal matrix where the ith diagonal element is the ith element of the vector v. We can rewrite (9) as

$$C_{A,A} = P_{A,A} + \mathrm{diag}(P_{A,B}e)Z, \tag{11}$$

where

$$Z = P^*_{A,B}\left[I - P_{B,B}\right]^{-1}P_{B,A}$$

and $P^*_{A,B}$ is simply $P_{A,B}$, but with all rows normalized to sum to 1. Let r_i be the ith diagonal element of $\mathrm{diag}(P_{A,B}e)$. The probabilistic interpretation of r_i is that it is the total probability of making a transition from state $s_i \in A$ to any state in B. Also, let z_i be the ith row of Z; then we can rewrite (11) as

$$C_{A,A} = P_{A,A} + \begin{bmatrix} r_1 z_1 \\ r_2 z_2 \\ \vdots \\ r_n z_n \end{bmatrix}. \tag{12}$$

Remarks. The probabilistic interpretation of (12) is as follows: If, in the original Markov chain, there is a transition from state $s_i \in A$ to any state in B, then, in the stochastic complement, this transition becomes a transition to some state(s) in A instead. In other words, the derived Markov chain "skips over" the period of time spent in B. The transition from $s_i \in A$ to B becomes a transition to $s_j \in A$ with probability z_{ij}. The stochastic complement of $P_{A,A}$ is therefore equal to $P_{A,A}$ plus any transition probabilities, which used to go from A to B, "folded" back to A and redistributed according to the stochastic matrix Z. This interpretation implies that the ith row of matrix Z determines how r_i should be redistributed back into A. In general, it is not an easy task to compute Z, but, for some special cases where sufficient "structure" exists in the original Markov chain, Z can be obtained with little or no computation. We end this section with a useful theorem.

Theorem 3 (Single entry). *Given an irreducible Markov process with state space S, let us partition the state space into two disjoint sets A and B. The transition rate matrix of this Markov process is*

$$\begin{bmatrix} Q_{A,A} & Q_{A,B} \\ Q_{B,A} & Q_{B,B} \end{bmatrix},$$

where $Q_{i,j}$ is the transition rate submatrix corresponding to transitions from partition i to partition j. If $Q_{B,A}$ has all zero entries except for some nonzero entries in the ith column, then the conditional steady state probability vector (corresponding to the states in A), given that the system is in partition A, is denoted by $\pi_{|A}$ and is the solution of the following system of linear equations:

$$\pi_{|A}\left[Q_{A,A} + Q_{A,B}ee_i^T\right] = \mathbf{0}$$
$$\pi_{|A}e = 1,$$

where e_i^T is a row vector with a 0 in each component, except for the ith component, which has the value of 1.

Proof. This is intuitively clear based on the stochastic complementation argument above. The matrix Z will have identical rows where each row is equal to e_i. This is true because, no matter how B is entered, A is entered from B via the ith state, with probability 1. □

4 General Approach

In this section, we describe a general approach to solving the model presented in Section 2. In particular, we first present an exact solution which exploits the matrix geometric structure described in Section 3.1. Because this approach suffers from a high computational cost, we then present an aggregation/disaggregation-based technique which results in significant computational savings (as shown later in the paper). We note that, due to a lack of structure in this model, we are not able to apply the aggregation/disaggregation technique directly; hence the need for a bounding approach. This bounding approach makes use of the aggregation/disaggregation technique and the matrix-geometric structure of the model; it is developed in detail in Sections 5 and 6.

4.1 Matrix Geometric Solution

The model presented in Section 2 is complex but has a special structure. If we partition the state space of $\mathcal{M}$ into disjoint sets B_i, $i \in \{0, 1, \cdots\}$, where

$$B_0 = \{(i,j,l) \in \mathcal{M} : i \leq F_{K-1} + 1\} \tag{13}$$

$$B_n = \{(i,j,K) \in \mathcal{S}_K : i = F_{K-1} + 1 + n\} \quad n \geq 1, \tag{14}$$

then $\mathcal{M}$ has a matrix geometric solution [30] as described in Section 3.1, where the components of each of the $\mathbf{A}_i$ submatrices are

$$\mathbf{A}_0[i,j] = \begin{cases} \lambda & \text{if } i = j \text{ and } 1 \leq i \leq K \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

$$\mathbf{A}_2[i,j] = \begin{cases} i\mu & \text{if } i = j \text{ and } 1 \leq i \leq K \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

$$\mathbf{A}_1[i,j] = \begin{cases} -[(K-i)\alpha + i\mu + \lambda] & \text{if } i = j \text{ and } 1 \leq i \leq K \\ (K-i)\alpha & \text{if } j = i+1 \text{ and } 1 \leq 1 \leq K-1 \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Solving a model using the matrix geometric method involves: 1) computing the $\mathbf{R}$ matrix, using the procedure in (3) and (4), for the repetitive part, and 2) solving the system of linear equations corresponding to the northwest corner of the generator matrix, i.e., solving $\mathbf{B}_{0,0}$, as defined by (2) and (13). As long as both matrices are "reasonably small," we could obtain an efficient solution. Computing the $\mathbf{R}$ matrix, i.e., Step 1) above, requires a computational cost of $\mathrm{O}(K^3)$. However, $\mathbf{B}_{0,0}$ is very large. The number of states corresponding to it is

$$1 + F_1 + \sum_{l=2}^{K-1} l(F_l - R_{l-1}) + K(F_{K-1} + 1 - R_{K-1}).$$

Given the original Markov process $\mathcal{M}$, let us partition the state space $\mathcal{S}$ into K disjoint sets $\mathcal{S}_l$, where

$$\mathcal{S}_l = \{(i,j,l) \mid (i,j,l) \in \mathcal{S} \text{ and } j \leq l\} \quad l = 0,..,K. \tag{18}$$

Then, for Step 2) above, the exact analysis requires a computational cost of $\mathrm{O}((n_0 + n_1 + n_2 + n_3 + \ldots + C_K)^3)$, where n_i is the dimension of $\mathcal{S}_i$, for $0 \leq i \leq K-1$, and C_K refers to the number of states in $\mathcal{S}_K$ with $F_{K-1} + 1$ or less customers (refer to Fig. 1).

One approach to reducing the computational complexity is to partition $\mathcal{M}$ into subparts, solve each subpart separately, and then combine the solutions, i.e., through the method of decomposition [3]. In our model, we can partition $\mathcal{M}$ into K subparts, each corresponding to set $\mathcal{S}_i$, for $0 \leq i \leq K$. Solving each $\mathcal{S}_i$ separately will allow us to lower the computational cost (the computational cost of solving the aggregate model is only $\mathrm{O}(K^3)$; see Section 4.3 for details). More specifically, the cost for solving all K subparts will be $\mathrm{O}((n_0)^3) + \mathrm{O}((n_1)^3) + \mathrm{O}((n_2)^3) + \ldots + \mathrm{O}(G_K)$, where G_K refers to the cost of solving $\mathcal{S}_K$. This is much smaller than the original computational cost if we can show that G_K is also "small."

Note that $\mathcal{S}_K$ is infinite, but also has a matrix geometric structure. That is, we can partition the state space of $\mathcal{S}_K$ into disjoint sets B_i, $i \in \{0, 1, \cdots\}$, where

$$B_0 = \{(i,j,K) \in \mathcal{S}_K : i \leq F_{K-1} + 1\} \tag{19}$$

$$B_n = \{(i,j,K) \in \mathcal{S}_K : i = F_{K-1} + 1 + n\} \quad n \geq 1, \tag{20}$$

and the matrices $\mathbf{A}_i$ are given in (15)-(17). We can then apply the solution of Section 3.1. In this case, the computational cost for computing $\mathbf{R}$ is also $\mathrm{O}(K^3)$ and, thus, $G_K = O((C_K)^3) + O(K^3)$.

For the applications where the number of servers, K, is large, we expect a significant improvement in computational complexity when using the decomposition method. What remains to be determined is whether it is possible to partition $\mathcal{M}$ and solve each $\mathcal{S}_i$ separately. This is the topic of the following section.

4.2 Partitioning of $\mathcal{M}$

If we could partition the state space of the original Markov process $\mathcal{M}$ into disjoint sets, then we could use stochastic complementation (see Section 3.2) to compute the conditional steady state probability vector for each set, given that the original Markov process $\mathcal{M}$ is in that set. By applying the state aggregation technique [3], we can aggregate each set into a single state and then compute the steady state probabilities for the aggregated process, i.e., the probabilities of the system being in any given set (see Section 4.3). Last, we can compute the individual (unconditional) steady state probabilities of the original Markov process $\mathcal{M}$ [3]; these can in turn be used to compute various performance measures. Unfortunately, the basic problem here is that we are not able to find special structure in the original model, such as the "single entry" structure exploited in Theorem 3, which can aid in determining the matrix Z (refer to (11)).

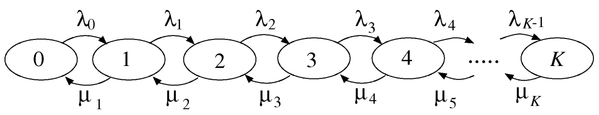


Fig. 2. State transition diagram for the aggregated process.

4.2.1 Alternative Approach

If, on the other hand, we could alter our model such that the single entry structure would exist, then we would be able to take advantage of Theorem 3 and, in essence, "disentangle" the partitions and solve each one separately. This would give us an approximate solution of the original model. If, in addition, we were able to alter the original model such that not only did we have the special structure, but were also able to obtain *provable* (performance) bounds, then we would also be able to bound the error due to this approximation. The bounding technique used in solving the model of Section 2 is given in Sections 5 and 6, where we illustrate how to construct and solve the upper and lower bound models as well as *prove* that these models do indeed provide bounds on the desired performance measures. Numerical results illustrating that 1) our bounds are *tight* and 2) the bounding technique results in *significant computational savings* are given in Section 8.

4.3 Analysis of the Aggregated Process

In this section, we briefly describe the analysis of the aggregated process, which is needed once the conditional steady state probabilities for the disjoint sets of states are computed. For each l, $1 \leq l \leq K$, we can aggregate all the states in $\mathcal{S}_l$ into a single state. The transition state diagram of the resulting aggregated process is illustrated in Fig. 2. The transition rates of the aggregated process can be computed as follows:

$$\lambda_0 = \lambda \tag{21}$$

$$\lambda_i = \lambda \sum_{j=1}^{i} \pi_i(F_i, j, i) \qquad i = 1, 2, \ldots, K-1 \tag{22}$$

$$\mu_i = \mu \sum_{j=1}^{i} j\pi_i(R_{i-1} + 1, j, i) \qquad i = 1, 2, \ldots, K, \tag{23}$$

where $\pi_i(F_i, j, i)$ and $\pi_i(R_{i-1} + 1, j, i)$ are the conditional steady state probabilities, conditioned on being in set $\mathcal{S}_i$. (We show how to obtain these, for each of the bounds, in

Sections 5 and 6.) The aggregated process depicted in Fig. 2 is a simple birth-death process and, hence, the corresponding steady state probabilities are very simple to compute (refer to [16]).

For each l, $0 \leq l \leq K$, we need to determine: 1) $\pi_l(i,j)$ the conditional state probabilities of all states in $\mathcal{S}_l$, given that the system is in $\mathcal{S}_l$, and 2) $\pi(l)$, the steady state probability of being in state l of the aggregated process. Then, the steady state probability of each individual state (i,j,l) in $\mathcal{M}$ can be expressed as

$$\pi(i,j,l) = \pi_l(i,j)\pi(l) \qquad \text{where} \quad (i,j,l) \in \mathcal{S}_l. \tag{24}$$

4.4 Computation of Performance Measures

In this section, we briefly discuss computation of performance measures for the model of Section 2. Given the steady state probabilities, we can compute various performance measures of interest. More specifically, we can compute many performance measures which can be expressed in the form of a Markov reward function, $\mathcal{R}$, where $\mathcal{R} = \sum_{i,j,l} \pi(i,j,l)R(i,j,l)$ and $R(i,j,l)$ is the reward for state (i,j,l). Two useful performance measures are the expected number of customers and the expected response time.

We can easily compute $N_{\mathcal{S}_l}$, the expected number of customers, given that the system is in $\mathcal{S}_l$, for $0 \leq l \leq K-1$, by expressing it as a Markov reward function, where $R(i,j,l) = i$ (and, then, the expected response time is computed using Little's result [22]). Computation of $N_{\mathcal{S}_K}$, the expected number of customers given that the system is in $\mathcal{S}_K$, is a bit more involved. Specifically,

$$\begin{aligned} N_{\mathcal{S}_K} &= \sum_{\forall (i,j,K) \in B_0} i\, \pi(i,j,K) + \sum_{j=1}^{\infty} (F_{K-1} + 1 + j)\boldsymbol{\pi}_j \boldsymbol{e} \\ &= \sum_{\forall (i,j,K) \in B_0} i\, \pi(i,j,K) + (F_{K-1}+1)(1 - \boldsymbol{\pi}_0 \boldsymbol{e}) \\ &\quad + \boldsymbol{\pi}_1 (\boldsymbol{I} - \mathbf{R})^{-2} \boldsymbol{e}. \end{aligned} \tag{25}$$

Note that $\pi(i,j,K)$ in the first summation term is simply one of the components of $\boldsymbol{\pi}_0$, which we defined in Section 3.1.

5 Upper Bound

In this section, we describe two models which can provide upper bounds on the desired performance measures for the model described in Section 2, namely, the mean number of customers and the mean system response time. For both upper bound models, we begin by illustrating the idea through the construction of these models. Then, for the first upper bound model, we present our proof of the fact that it does indeed result in an upper bound on performance metrics of interest. Last, we give a computational procedure for obtaining the desired performance measures. For the second upper bound, we omit the proof and the computational procedure as they are similar in nature to the proof and computational procedure of the first upper bound model.

5.1 Upper Bound Model

The basic idea for the construction of the upper bound model, $\mathcal{M}_1^u$, is as follows: We alter several transitions in the original model while satisfying the criteria that the new model will: 1) provide a (hopefully tight) upper bound on the desired performance measures and 2) be a *simpler* model to solve. As pointed out in Section 4, we would like to solve this model using the decomposition method. The difficulty with applying this approach to the original model is the existence of multiple entry states in $\mathcal{S}_l$, from both $\mathcal{S}_{l-1}$ and $\mathcal{S}_{l+1}$. Thus, we will construct the upper bound model by altering transitions in the original model and creating a single entry state "somewhere" in $\mathcal{S}_l$. Intuitively, we will be modifying the departure processes, as compared to the original model, such that $\mathcal{M}_1^u$ will have fewer active servers. That is, $\mathcal{M}_1^u$ and $\mathcal{M}$ will "see" the same arrivals, but, at any given moment, $\mathcal{M}_1^u$ will have the same or *fewer* number of servers processing these arrivals. Note that these judicious modifications of the departure process will allow us to have a tight upper bound as well as an efficient computational procedure (see Section 8).

We begin at the lowest level—for instance, in the case of Fig. 1, we begin at level $\mathcal{S}_3$. To achieve the upper bound, we can alter the following transitions (which are indicated by dashed transition lines in Fig. 3). The original transition from state $(R_2+1,2,3)$ to state $(R_2,2,2)$ is changed to a transition to state $(R_2,1,2)$, at the same rate. In addition, the original transition from state $(R_2+1,3,3)$ to state $(R_2,2,2)$ is changed to a transition to state $(R_2,1,2)$, at the same rate.

In general, we can describe the upper bound version of our model, $\mathcal{M}_1^u$, as follows: We can construct a corresponding Markov process, $\mathcal{M}_1^u$, with the following state space $\mathcal{S}^{u1}$:

$$\begin{aligned} \mathcal{S}^{u1} = \{ (N_1^u, N_{s1}^u, L_1^u) \mid N_1^u \geq 0, \\ N_{s1}^u \in \{0,1,2,\ldots,K\}, L_1^u \in \{0,1,2,\ldots,K\} \}, \end{aligned}$$

where N_1^u is the number of customers in the queuing system, N_{s1}^u is the number of activated servers, and L_1^u is the level to which the state belongs (see Section 2 for an explanation of the "level" notation). The transition structure of $\mathcal{M}_1^u$ is the same as that of the original process $\mathcal{M}$, given in (1), except for the transition corresponding to a change of levels due to a departure (line 5 in (1)). The upper bound model transitions that replace this original transition can be specified as follows:

$$\begin{aligned} &(i,j,l) \quad \longrightarrow \quad (i-1, \min(j, l-1), l-1) \\ &j\mu \mathbf{1}\{ \ (i-1 = R_k \in \boldsymbol{R}) \wedge (l = k+1) \wedge (l \leq 2) \ \} \\ &(i,j,l) \quad \longrightarrow \quad (i-1, 1, l-1) \\ &j\mu \mathbf{1}\{ \ (i-1 = R_k \in \boldsymbol{R}) \wedge (l = k+1) \wedge (l \geq 3) \ \}, \end{aligned} \tag{26}$$

where $\mathbf{1}\{x\}$ is an indicator function of x.

5.1.1 *Second Upper Bound Model*

An alternative upper bound model can be constructed by considering *arrivals* rather than *departures*, as was done above. As in the first upper bound model, we strive to satisfy the criteria that the new model will: 1) provide a (hopefully tight) upper bound on the desired performance measures and 2) be a "simpler" model to solve. Again, the alterations of transitions are done to create a single entry

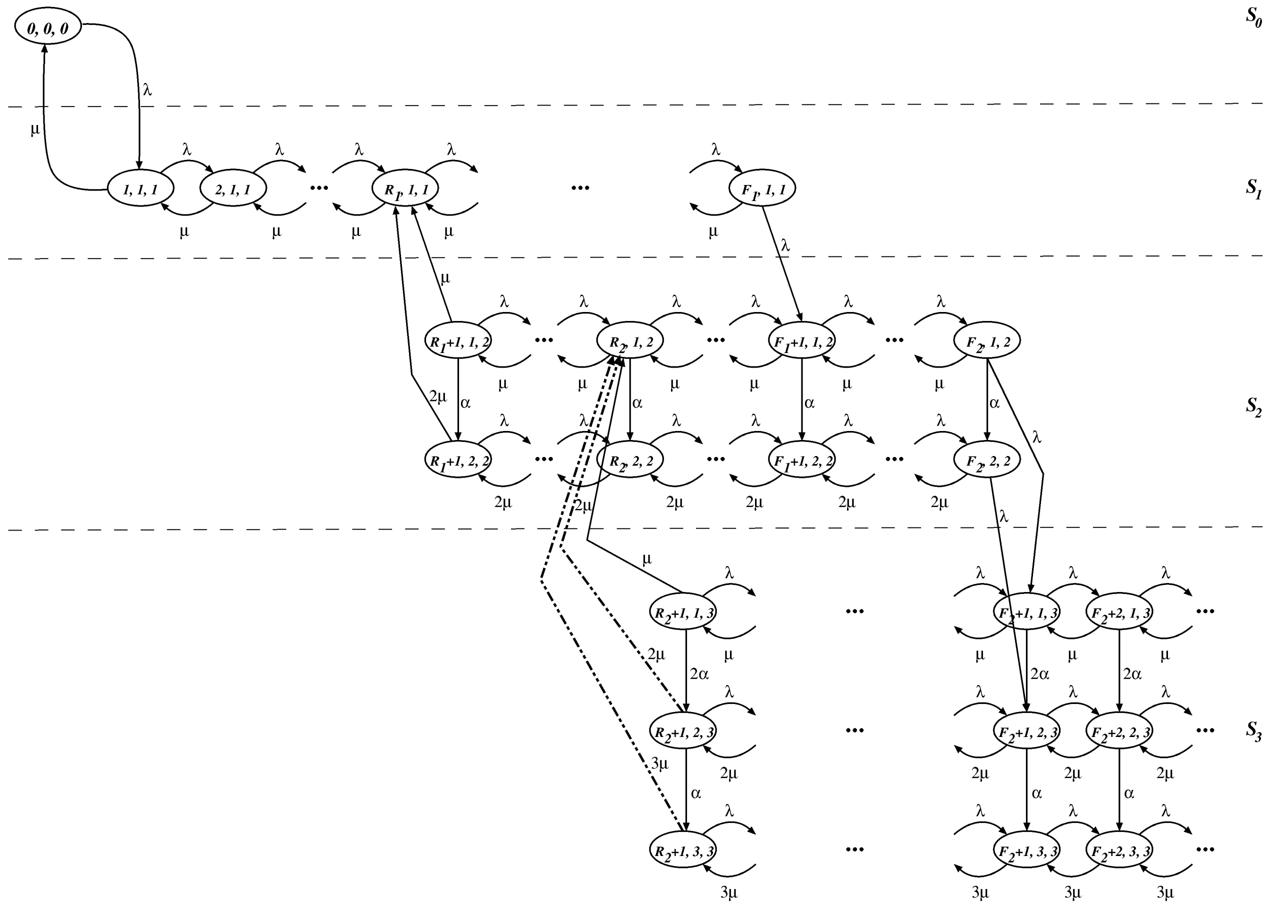


Fig. 3. State transition diagram for the upper bound $\mathcal{M}_1^u$.

state "somewhere" in $\mathcal{S}_l$, but, instead of altering departure-related transitions, we alter arrival-related transitions. Intuitively, we will be modifying the arrival processes, as compared to the original model, such that $\mathcal{M}_2^u$ will have fewer active servers. That is, $\mathcal{M}_2^u$ and $\mathcal{M}$ will "see" the same arrivals, but, at any given moment, $\mathcal{M}_2^u$ will have the same or *fewer* number of servers processing these arrivals. Note that these judicious modifications of the arrival process will allow us to have a tight upper bound as well as an efficient computational procedure.

The transition structure of $\mathcal{M}_2^u$ is the same as that of the original process $\mathcal{M}$, given in (1), except for the transition corresponding to a change of levels due to an arrival (line 2 in (1)). The upper bound model transitions that replace this original transition can be specified as follows:

$$\begin{array}{lll} (i,j,l) & \longrightarrow & (i+1,j,l+1) \\ \lambda \mathbf{1}\{ \ (i = F_k \in \boldsymbol{F}) \wedge (l = k) \wedge (l \leq 1) \ \} & & \\ (i,j,l) & \longrightarrow & (i+1,1,l+1) \\ \lambda \mathbf{1}\{ \ (i = F_k \in \boldsymbol{F}) \wedge (l = k) \wedge (l \geq 2) \ \}. & & \end{array} \quad (27)$$

An example of $\mathcal{M}_2^u$ is illustrated in Fig. 4, where the modified transitions (as compared to the original model of Fig. 1) are indicated by dashed lines.

The proof that $\mathcal{M}_2^u$ provides an upper bound on the desired performance measures is similar to the proof given in Section 5.2 in the context of $\mathcal{M}_1^u$. Likewise, the numerical computation of performance measures using $\mathcal{M}_2^u$ is similar to the numerical computations given in Section 5.3 in the context of $\mathcal{M}_1^u$.

5.2 Proof for the Upper Bound Model $\mathcal{M}_1^u$

In this section, we give a theorem and the corresponding proof that $\mathcal{M}_1^u$ provides an upper bound on the mean number of customers and the mean response time of the original model $\mathcal{M}$. Using the state notation defined in the previous section, we have the following notation for the original model $\mathcal{M}$, $[N(t), N_s(t), L(t)]$, where $N(t)$ is the number of jobs waiting in the queue at time t, $N_s(t)$ is the number of activated servers at time t (where $0 \leq N_s(t) \leq K$), and $L(t)$ is the level to which the state belongs at time t. Similarly, we define the state vector for the upper bound model $\mathcal{M}_1^u$ as $[N_1^u(t), N_{s1}^u(t), L_1^u(t)]$.

We first give a definition of stochastic comparison [25] that will be useful in our proof and then state the theorem.

Definition 2. *Let X and Y be two real valued random variables. X is stochastically less than Y ($X \leq_{st} Y$) iff $P[Y < t] \leq P[X < t]$ $\forall t$.*

Theorem 4. *If $N(0) \leq_{st} N_1^u(0)$, then we have $N(t) \leq_{st} N_1^u(t)$ $\forall t$.*

Proof. We shall use a sample path analysis argument [24], [33] to show our result. Below, we compare the two systems on a specific sample path (which is accomplished through coupling [33]). If we can show that the above-stated relationship holds on a "generic" sample

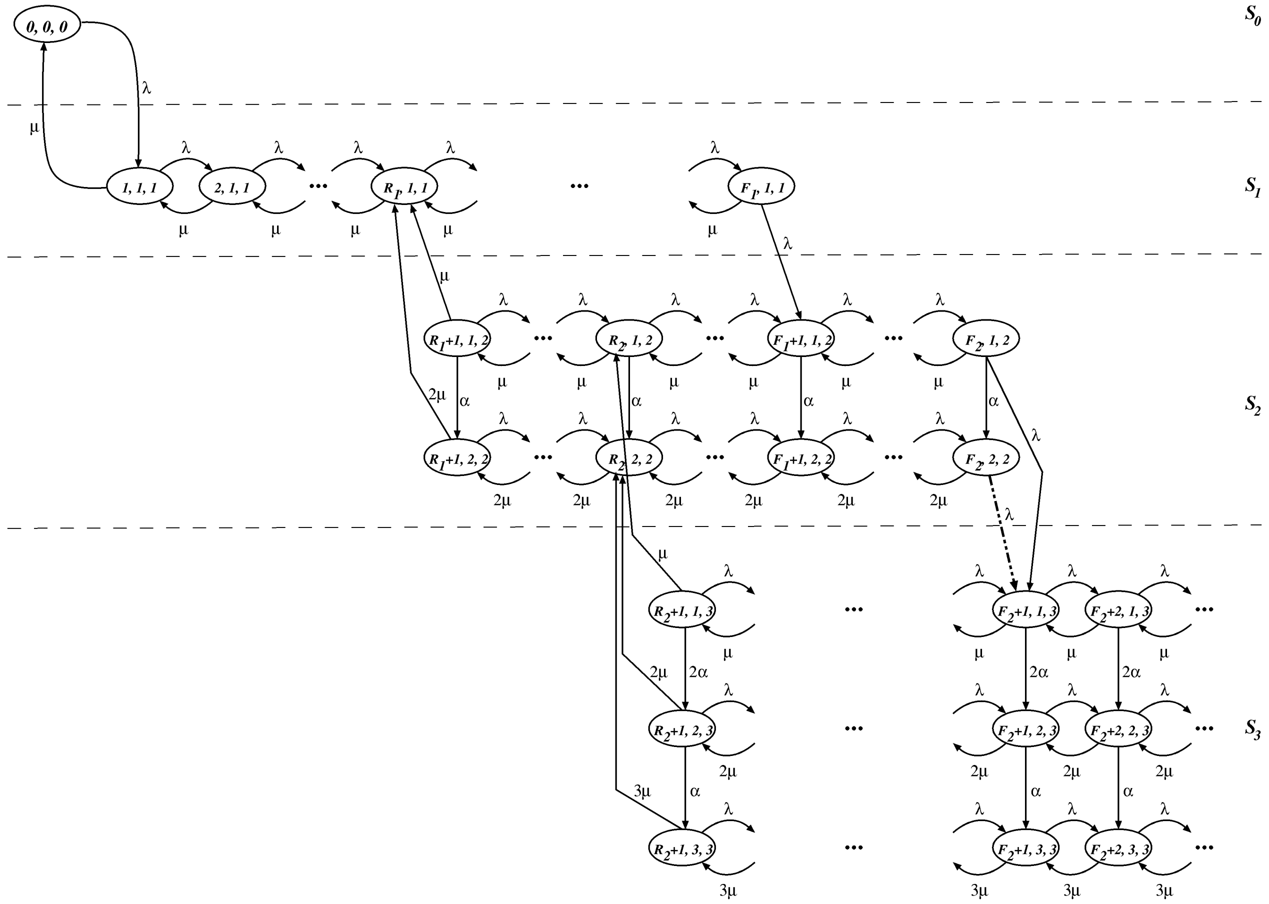


Fig. 4. State transition diagram for the upper bound $\mathcal{M}_2^u$.

path, then it follows that the corresponding stochastic relationship holds as well. We couple [21] the initial total number of customers in both systems such that $N(0) \leq N_1^u(0)$ and condition on the three possible events that can occur in these systems, namely: 1) an arrival event, 2) a server activation event, and 3) a service event. We also couple the two systems ($\mathcal{M}$ and $\mathcal{M}_1^u$) through the arrival process, i.e., whenever there is an arrival to one system, there is also an arrival to the other system. Finally, recall that the exponential assumptions (for 1) server activation and 2) service times) guarantee that the remaining time to the next server activation or the next service event is also an exponential random variable with the *same* parameter.

Let $\{t_n\}$ be a sequence of times where each t_i corresponds to the *i*th system event (i.e., either arrival or server activation or service). We then prove the above statement by induction.

Basis step. For $t_0 = 0$, the result follows from the initial coupling of these two variables and, therefore, $N(0) \leq N_1^u(0)$.

Inductive step. Assume that the hypothesis holds for the first n events. We then show that it also holds for the $(n+1)$st event by distinguishing each type of an event as follows:

- *Arrival event:* In the original system $\mathcal{M}$, we have $N(t_{n+1}) = N(t_n) + 1$. Since we do not reject any arrivals in the upper bound model $\mathcal{M}_1^u$, we also have $N_1^u(t_{n+1}) = N_1^u(t_n) + 1$ and, therefore, $N(t_{n+1}) \leq N_1^u(t_{n+1})$.
- *Server activation event:* During any server activation event, the total number of customers in the system remains the same in both the original model $\mathcal{M}$ and the upper bound model $\mathcal{M}_1^u$. Therefore, $N(t_{n+1}) \leq N_1^u(t_{n+1})$.
- *Service event:* To show that the stochastic relation holds between the two models, we perform the following "pairing" of servers. We number the (homogeneous) servers in both systems, from 1 to K. When there is a departure from server j in one system **and** server j in the other system is *busy*, we force a departure from server j in the other system as well. (Note that this is another legitimate form of coupling due to the assumption of exponential service times.)

Let $K_{orig}(t)$ ($K_u(t)$) be the number of *activated* servers at time t in model $\mathcal{M}$ ($\mathcal{M}_1^u$). We apply the following mapping in both models. If the number of customers $N(t) < K_{orig}(t)$ in model $\mathcal{M}$, then these customers reside on the "first" $N(t)$ servers. Similarly, if there are $N_1^u(t) < K_u(t)$ customers in model $\mathcal{M}_1^u$, then they also reside on the "first" $N_1^u(t)$ servers. Hence, if a departure occurs from some server other than $N(t)$ (similarly, $N_1^u(t)$), we just "reshuffle" the customers to satisfy this constraint. (Again, this is a legitimate operation due to the assumption of exponential service times, i.e., we can "move" a customer from one server to another and resample its service time, because the remaining service time of a customer in our system is an exponentially distributed random variable with the *same* parameter as its original service time distribution).

We distinguish between three possible cases of the service event, depending on whether we have a busy or an idle server in each of the two systems. That is, a departure from server j occurs in one of the systems and we consider how this event affects both systems.

1. Server j is busy in both systems and, thus, both systems experience a departure of one customer. Then, clearly, we have $N(t_{n+1}) \leq N_1^u(t_{n+1})$.
2. Server j is busy in $\mathcal{M}$ but not in $\mathcal{M}_1^u$, thus there is a departure in $\mathcal{M}$ only. The fact that $N(t_n) \leq N_1^u(t_n)$ before this event means that server j has not yet been activated in $\mathcal{M}_1^u$. (Recall that the number of activated servers in $\mathcal{M}$ is at least as high as in $\mathcal{M}_1^u$).) Thus, the relationship is still maintained and we have $N(t_{n+1}) < N(t_n) \leq N_1^u(t_n) = N_1^u(t_{n+1})$, i.e., after this event $N(t_{n+1})$ is *strictly less* than $N_1^u(t_{n+1})$.
3. Server j is busy in $\mathcal{M}_1^u$ but not in $\mathcal{M}$ and, thus, there is a departure in $\mathcal{M}_1^u$ only. The fact that $N(t_n) \leq N_1^u(t_n)$ before this event means that server j is *not busy* in $\mathcal{M}$, i.e., in fact $N(t_n) < N_1^u(t_n)$ **and** $N(t_n) < K_{orig}(t_n)$ before t_{n+1}. Hence, the relationship is maintained and we have $N(t_{n+1}) \leq N_1^u(t_{n+1})$.

By removing conditions on the initial state of the system, arrival times, server activation times, and service event times, we have that $N(t) \leq_{st} N_1^u(t)\ \forall t$. □

Let R and R_1^u be the random variable denoting the response time of customers in models $\mathcal{M}$ and $\mathcal{M}_1^u$, respectively. Then, we have the following corollary.

Corollary 1. $E[R] \leq E[R_1^u]$.

Proof. Let $N = \lim_{t\to\infty} N(t)$ and $N_1^u = \lim_{t\to\infty} N_1^u(t)$ (note that the limit's existence in the distributional sense is justified because of the geometric tail in the models). Based on the previous theorem, we have $E[N] \leq E[N_1^u]$. Since both systems are subjected to the same external arrival process with rate λ, by Little's result [22], we have $E[R] \leq E[R_1^u]$. □

5.3 Numerical Computation for Model $\mathcal{M}_1^u$

In this section, we present a numerical computation procedure for the upper bound model $\mathcal{M}_1^u$. Our goal here will be to partition the upper bound model into disjoint sets and apply the stochastic complementation approach of Section 3.2. Let us first define the following notation:

$$\mathcal{S}_l^- = \bigcup_{i=0}^{l} \mathcal{S}_i \quad \text{and} \quad \mathcal{S}_l^+ = \bigcup_{i=l}^{K} \mathcal{S}_i,$$

where (for ease of notation) S_i, $0 \leq i \leq K$, is defined as in (18), but with respect to the upper bound model $\mathcal{M}_1^u$. Then, we can state the following theorem.

Theorem 5 (Multiple entries). *Given an irreducible Markov process, $\mathcal{M}_1^u$, of Section 5.1, with state space $\mathcal{S}^{u1}$, let us partition the state space into two disjoint sets $\mathcal{S}_{l-1}^-$ and $\mathcal{S}_l^+$, for $3 \leq l \leq K$. The transition rate matrix of this Markov process is:*

$$\begin{bmatrix} \boldsymbol{Q}_{\mathcal{S}_l^+,\mathcal{S}_l^+} & \boldsymbol{Q}_{\mathcal{S}_l^+,\mathcal{S}_{l-1}^-} \\ \boldsymbol{Q}_{\mathcal{S}_{l-1}^-,\mathcal{S}_l^+} & \boldsymbol{Q}_{\mathcal{S}_{l-1}^-,\mathcal{S}_{l-1}^-} \end{bmatrix},$$

where $\boldsymbol{Q}_{i,j}$ is the transition rate submatrix corresponding to transitions from partition i to partition j. Then, r_k and z_k, $1 \leq k \leq n$, in (12) are as follows:

$$r_k = \begin{cases} j\mu & if\ k = (R_{l-1}+1, j, l)\ and\ 1 \leq j \leq l \\ 0 & otherwise \end{cases}$$

$$z_{k,n} = \begin{cases} \frac{\pi_{l-1}((F_{l-1},x,l-1))q_{(F_{l-1},x,l-1),(F_{l-1}+1,x,l)}}{\sum_{y=1}^{l-1}\pi_{l-1}((F_{l-1},y,l-1))q_{(F_{l-1},y,l-1),(F_{l-1}+1,y,l)}} \\ if\ k = (R_{l-1}+1, j, l), 1 \leq j \leq l\ and \\ n = (F_{l-1}+1, x, l)\ and\ 1 \leq x \leq l-1 \\ 0 \qquad otherwise, \end{cases}$$

where $\pi_l(n)$ is the probability of being in state n conditioned on being in $\mathcal{S}_l$ and $q_{i,j}$ is the transition rate from state i to state j in $\mathcal{M}_1^u$. Then, the stationary state probability vector, $\boldsymbol{\pi}_{|\mathcal{S}_l^+}$ is given by (10).

Proof. This follows from the definition of a stochastic complement. □

A very similar theorem can be stated for constructing a stochastic complement for $Q_{\mathcal{S}_l^-,\mathcal{S}_l^-}$; we omit it in the interests of brevity. Given these theorems, we can construct the stochastic complement for each set $\mathcal{S}_l$ and compute the conditional steady state probabilities.

Let us now present the numerical computation procedure for the upper bound model $\mathcal{M}_1^u$. The basic idea is that we first compute the steady state probability vector given that the system $\mathcal{M}_1^u$ is in a particular set, namely, $\mathcal{S}_l$ for $0 \leq l \leq K$ (this procedure is described below). Then, we compute the aggregate state probabilities for each $\mathcal{S}_l$ as described in Section 4.3. Based on these two values, we can compute the individual steady state probabilities as well as the desired performance measures.

We now concentrate on computation of the steady state probabilities given that the system $\mathcal{M}_1^u$ is in a particular set $\mathcal{S}_l$. Let $N_{\mathcal{S}_l}$ denote the expected number of customers, given that the system is operating in $\mathcal{S}_l$, $0 \leq l \leq K$. For $\mathcal{S}_0$, the conditional steady state probability is clearly equal to 1 and $N_{\mathcal{S}_0} = 0$. For $\mathcal{S}_1$, there is a single exit state to $\mathcal{S}_i$ and a single entry state from $\mathcal{S}_i$, where $i \in \{0, 2\}$. Therefore, we can apply Theorem 3 twice, fold the transition from state $(F_1, 1, 1)$ to state $(R_1, 1, 1)$ with rate λ and from state $(1, 1, 1)$ back to $(1, 1, 1)$ with rate μ and compute the steady state probability vector, given that the system is in $\mathcal{S}_1$. Once this is obtained, $N_{\mathcal{S}_1}$ is easily computed.

Level $\mathcal{S}_2$ has a single entrance state from level $\mathcal{S}_3$ and also a single entrance state from level $\mathcal{S}_1$. Therefore, we can apply Theorem 3 again and fold both transitions from states $(F_2, 1, 2)$ and $(F_2, 2, 2)$ to state $(R_2, 1, 2)$, each at the rate of λ. In addition, we can fold both transitions from states $(R_1 + 1, 1, 2)$ and $(R_1 + 1, 2, 2)$ to state $(F_1 + 1, 1, 2)$, the former at the rate of μ and the latter at the rate of 2μ. At this point, we can compute the steady state probability vector, given that the system is in $\mathcal{S}_2$, using a variety of methods (refer to [32]). Once this is done, we can easily compute $N_{\mathcal{S}_2}$.

For level $\mathcal{S}_l$, where $3 \le l \le K-1$, we first note that there is a single entry state from the states in $\mathcal{S}_{l+1}$. Therefore, using Theorem 3, we can fold the transitions from state (F_l, j, l), where $1 \le j \le l$, to state $(R_l, 1, l)$, each with a rate equal to λ. On the other hand, there are multiple exit states to $\mathcal{S}_{l-1}$ and multiple entry states from $\mathcal{S}_{l-1}$ to $\mathcal{S}_l$. Since we have computed the conditional steady state probabilities, given that the system is in $\mathcal{S}_{l-1}$ in a previous step, using Theorem 5, we can determine exactly how to fold these transitions from the exit states $(R_{l-1} + 1, j, l)$ back to the entry states $(F_l + 1, j', l)$. Then, we can compute the conditional steady state probabilities, given that the system is in $\mathcal{S}_l$, using a variety of methods (refer to [32]). Once the conditional steady state probability vector is determined, we can easily compute $N_{\mathcal{S}_l}$, for $3 \le l \le K-1$.

The computation of the conditional steady state probabilities for set $\mathcal{S}_K$ is somewhat different. First, observe that we can apply Theorem 5 to fold the transitions from the exit state $(R_{K-1} + 1, j, K)$ back to the entry states $(F_{K-1} + 1, j', K)$. Since the state space cardinality of $\mathcal{S}_K$ is infinite, we cannot use standard numerical methods (such as the power method) to compute the conditional steady state probabilities in this case. However, since the Markov process corresponding to $\mathcal{S}_K$ has special structure, i.e., a quasi-birth-death version of the matrix geometric form, the remainder of the solution can proceed as described in Section 4. Note that, although solving the entire model using the approach of Section 4 is not computationally efficient, exploiting the matrix geometric form for the solution for $\mathcal{S}_K$ is computationally efficient since, in this case, $\mathbf{B}_{0,0}$ is relatively small, on the order of $(F_{K-1} - R_{K-1}) \times K$, and $\mathbf{R}$ is also relatively small, on the order of $K \times K$.

6 Lower Bound

In this section, we describe two models which can provide lower bounds on the desired performance measures for the model described in Section 2, namely, the mean number of customers and the mean system response time. The intuition behind the construction of the lower bound models is similar to that of the upper bound models. We alter several transitions in the original model while satisfying the criteria that the new model will: 1) provide a (hopefully tight) lower bound on the desired performance measures and 2) be a "simpler" model to solve.

6.1 First Lower Bound Model

As pointed out in Section 4, we would like to solve our model using the method of decomposition. Recall that the difficulty with applying this approach to the original model was the existence of multiple entry states in $\mathcal{S}_l$, from both $\mathcal{S}_{l-1}$ and $\mathcal{S}_{l+1}$. Thus, we will construct the lower bound model by altering transitions in the original model and creating a single entry state "somewhere" in $\mathcal{S}_l$. Intuitively, we will be modifying the departure process, as compared to the original model, such that $\mathcal{M}_1^l$ will have more active servers. That is, $\mathcal{M}_1^l$ and $\mathcal{M}$ will "see" the same arrivals, but, at any given moment, $\mathcal{M}_1^l$ will have the same or *greater number of* servers processing these arrivals. Note that these judicious modifications of the departure process will allow us to have a tight lower bound as well as an efficient computational procedure (see Section 8).

We begin at the lowest level, for instance, in the case of Fig. 1, we begin at level $\mathcal{S}_3$. To achieve the lower bound, we can alter the following transition (this is indicated by the dashed transition line in Fig. 5). The transition from state $(R_2 + 1, 1, 3)$ to state $(R_2, 1, 2)$ is changed to a transition to state $(R_2, 2, 2)$, at the same rate.

In general, we can describe the lower bound version of our model, $\mathcal{M}_1^l$, as follows: We can construct a corresponding Markov process, $\mathcal{M}_1^l$, with the following state space $\mathcal{S}^{l1}$:

$$\mathcal{S}^{l1} = \{(N_1^l, N_{s1}^l, L_1^l) \mid N_1^l \ge 0, N_{s1}^l \in \{0, 1, 2, \dots, K\}, \\ L_1^l \in \{0, 1, 2, \dots, K\}\},$$

where N_1^l is the number of customers in the queuing system, N_{s1}^l is the number of activated servers, and L_1^l is the level to which the state belongs (see Section 2 for explanation of the "level" notation). The transition structure of $\mathcal{M}_1^l$ is the same as that of the original process $\mathcal{M}$, given in (1), except for the transition corresponding to a change of levels due to a departure (line 5 in (1)). The lower bound model transitions that replace this original transition can be specified as follows:

$$\begin{aligned} &(i, j, l) \quad \longrightarrow \quad ; (i-1, \min(j, l-1), l-1) \\ &j\mu\mathbf{1}\{(i-1 = R_k \in \boldsymbol{R}) \wedge (l = k+1) \wedge (l \le 2)\} \\ &(i, j, l) \quad \longrightarrow \quad (i-1, l-1, l-1) \\ &j\mu\mathbf{1}\{(i-1 = R_k \in \boldsymbol{R}) \wedge (l = k+1) \wedge (l \ge 3)\}, \end{aligned} \tag{28}$$

where $\mathbf{1}\{x\}$ is an indicator function of x.

6.2 Second Lower Bound Model

An alternative lower bound model can be constructed by considering *arrivals* rather than *departures*. As in the first lower bound model, we strive to satisfy the criteria that the new model will: 1) provide a (hopefully tight) lower bound on the desired performance measures and 2) be a "simpler" model to solve. Again, the alterations of transitions are done to create a single entry state "somewhere" in $\mathcal{S}_l$, but, instead of altering departure-related transitions, we alter arrival-related transitions. Intuitively, we will be modifying the arrival process, as compared to the original model, such that $\mathcal{M}_2^l$ will have more active servers. That is, $\mathcal{M}_2^l$ and $\mathcal{M}$ will "see" the same arrivals, but, at any given moment, $\mathcal{M}_2^l$ will have the same or *greater number of* servers processing these arrivals. Note that these judicious modifications of the arrival process will allow us to have a tight lower bound as well as an efficient computational procedure.

The transition structure of $\mathcal{M}_2^l$ is the same as that of the original process $\mathcal{M}$, given in (1), except for the transition

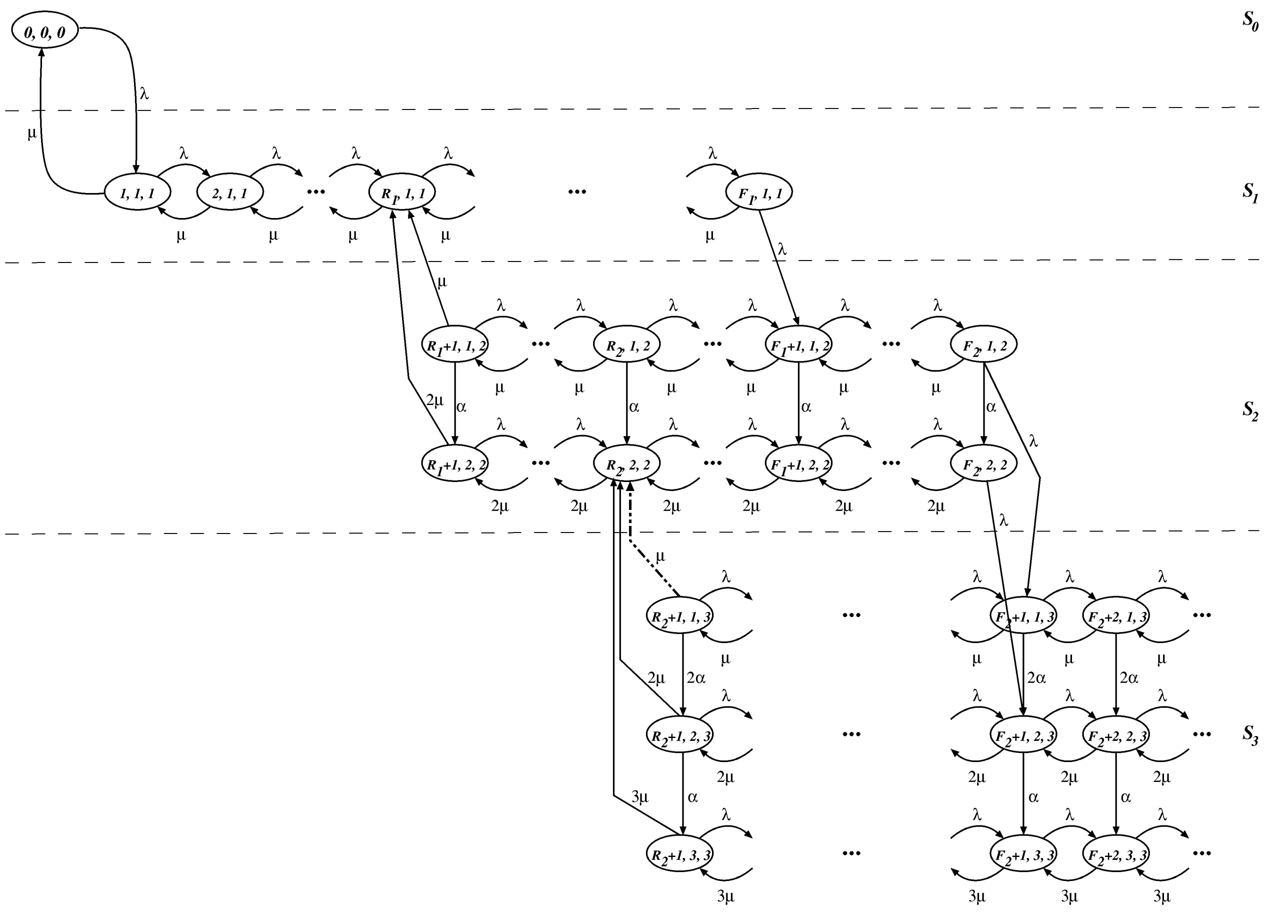


Fig. 5. State transition diagram for the lower bound model $\mathcal{M}_1^l$.

corresponding to a change of levels due to an arrival (line 2 in (1)). The lower bound model transitions that replace this original transition can be specified as follows:

$$\begin{aligned}&(i,j,l) \quad \longrightarrow \quad (i+1,j,l+1)\\ &\lambda\mathbf{1}\{(i=F_k\in \boldsymbol{F})\wedge(l=k)\wedge(l\le 1)\}\\ &(i,j,l) \quad \longrightarrow \quad (i+1,l,l+1)\\ &\lambda\mathbf{1}\{(i=F_k\in \boldsymbol{F})\wedge(l=k)\wedge(l\ge 2)\}.\end{aligned} \tag{29}$$

An example of $\mathcal{M}_2^l$ is illustrated in Fig. 6, where the modified transitions (as compared to the original model of Fig. 1) are indicated by dashed lines.

The proof that $\mathcal{M}_1^l$ and $\mathcal{M}_2^l$ provide lower bounds on the desired performance measures is similar to the proof given in Section 5.2 in the context of $\mathcal{M}_1^u$. Likewise, the numerical computation of performance measures using $\mathcal{M}_1^l$ or $\mathcal{M}_2^l$ is similar to that of Section 5.3 in the context of $\mathcal{M}_1^u$.

7 Application to VOD Servers

In this section, we illustrate the utility of threshold-based queuing models with hysteresis in the design of distributed systems and, specifically, in the design of multimedia systems. With technological advances in information and communication technologies, multimedia on-demand systems now play a major role in educational applications, entertainment technology, and library information systems. In this paper, we use the Video-on-Demand (VOD) systems as an example application. (For a survey on the design of a VOD system, we refer the reader to [5].) Due to the enormous storage and bandwidth requirements of multimedia data, such systems are expected to have very large disk farms. Thus, it would be unrealistic to consider a centralized design of a video server using a single disk cluster and/or a single processing node. One difficulty in designing any large distributed information system is the choice of data placement techniques. The distribution of data among the nodes of the system can significantly affect the overall performance of that system. Inappropriate data distribution can lead to load imbalance problems due to skewness in the data access patterns. Specifically, in a large distributed VOD system, improper data distribution can lead to a situation where requests for (popular) objects cannot be serviced, even when the overall capacity of the system is not exhausted, because these objects reside on highly loaded nodes, i.e., the available capacity and the necessary data are not on the same node.

In the recent past, a great deal of CM server designs, e.g., as in [1], [4], [9], have focused on "wide" data striping, where each object is striped across all the disks of the system. This approach "spreads" the load more or less evenly across all disks and avoids the problem of "partitioning resources," e.g., as in [1], which is illustrated in Fig. 7a, where object A is striped across N disks, i.e., its first block, A_1, is placed on the first disk, its second block, A_2 is placed on the second disk, and so on (to simplify the

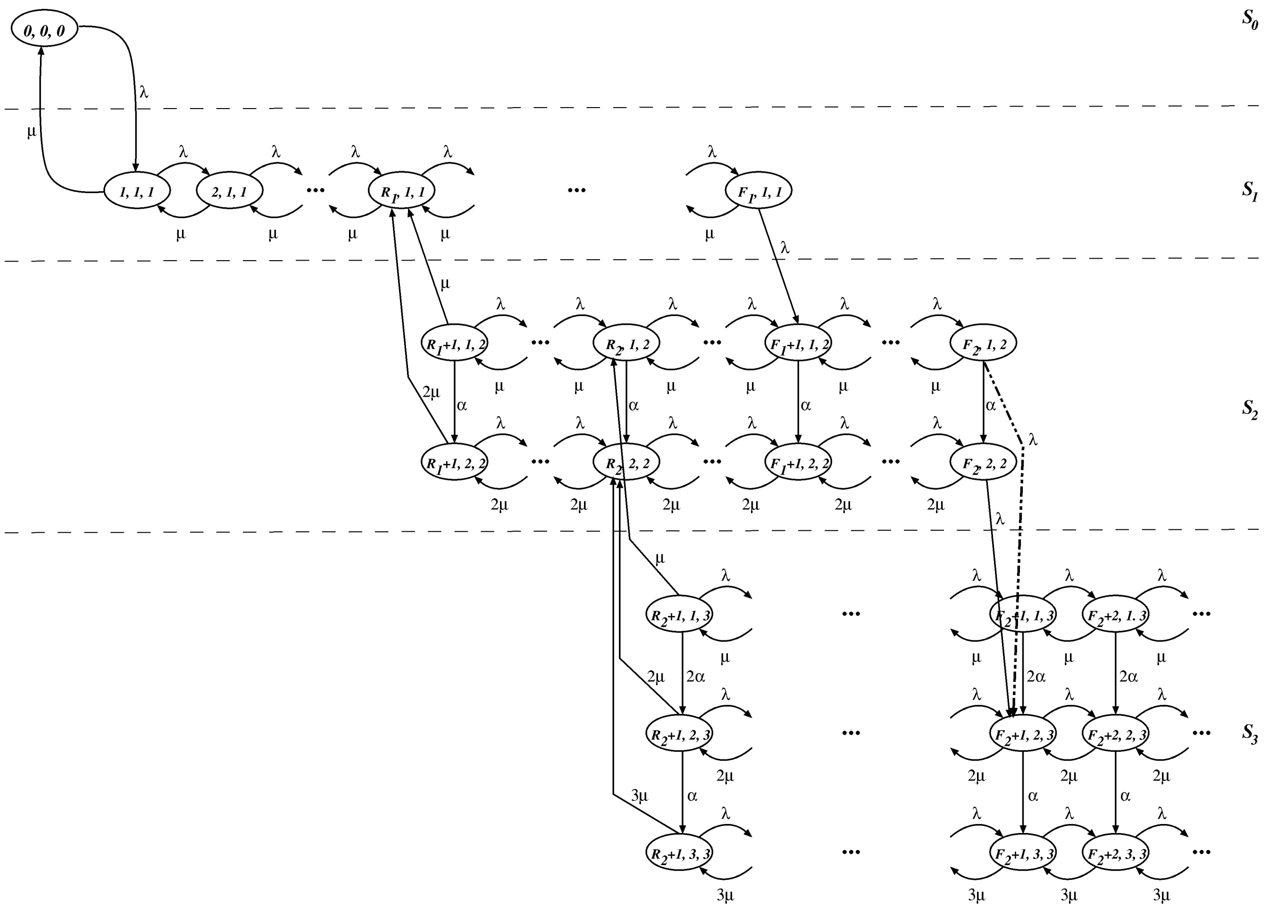
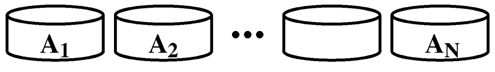


Fig. 6. State transition diagram for the lower bound $\mathcal{M}_2^l$.

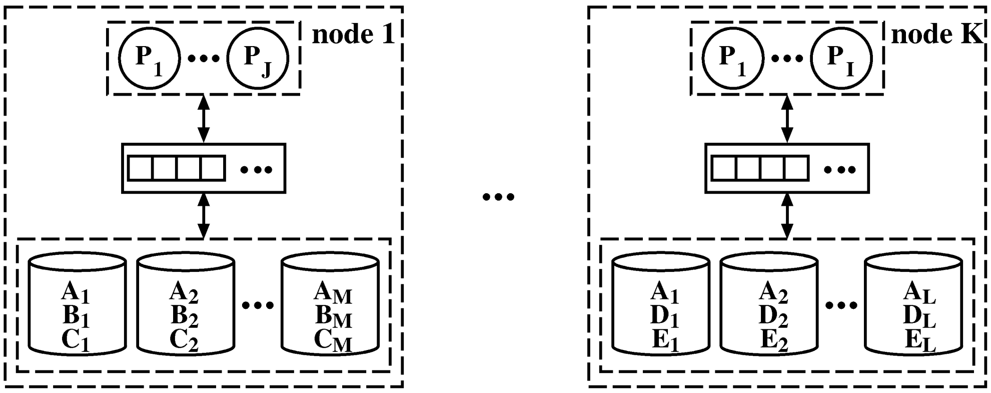
figure, we only illustrate the disks in this case, i.e., rather than also illustrating the processing nodes, the buffer space etc., as in Fig. 7b). However, this approach suffers from the following shortcomings: A processing node can only be attached to a limited number of disks, therefore, a multinode system must be considered and, thus, some form of synchronization in delivery of a single object from multiple nodes must be addressed (depending on the system and network architecture, this can be a significant problem). In addition, it is not practical to assume that a system can be constructed from homogeneous disks, i.e., as the system grows, we would be forced to use disks with different transfer and storage capacity characteristics. Having to stripe objects across heterogeneous disks would lead to further complications [19], [31]. Moreover, an appropriate choice of a striping unit and the communications network infrastructure dictate an upper bound on the number of disks over which an object can be striped, beyond which replication of objects is needed to increase the number of simultaneous users [4]. Finally, due to the potential need for communication of video data between the nodes over which the data is striped, the capacity of the communication network limits the performance of the distributed VOD server. This limitation directly affects the scalability of the VOD server, as shown in [2].

Instead of striping each objects across *all* the nodes of the system, we can take a *hybrid* approach, as in [34], and constrain the striping to a single node and replicate popular objects on several nodes in order to provide sufficient bandwidth capacity to serve the demand for these objects. This is illustrated in Fig. 7b, where each objects resides completely on a single node, but multiple copies of the same object may exist, e.g., there are two copies of object A. That is, one could take advantage of load balancing properties of striping by allowing an object to be striped *within* the disks of the same node, as illustrated in Fig. 7b, but avoid the shortcomings of striping by replicating across nodes [2].

In order to achieve better load balancing characteristics, the number of replicas of each object should change as the object access patterns change. Thus, another approach to dealing with skews and changes in data access patterns is to replicate objects *dynamically*, as demand for it arises. (For instance, in [2], we showed that hybrid designs, in conjunction with dynamic replication techniques, are less dependent on interconnection network constraints, provide higher reliability, and can be properly sized so as to result in cost-effective end-to-end systems.) Hence, dynamic replication of objects in distributed VOD systems is the focus of our analysis. Note that we do not advocate a particular approach to designing VOD servers, but rather show how our modeling methodology can be applied to one viable approach to VOD design (which, e.g., is taken in [34]).



(a)



(b)

Fig. 7. Data placement. (a) Striping. (b) Replication.

Since dynamic replication could result in significant system overheads [18], e.g., in the form of additional I/O disk retrievals, memory buffers, and communication bandwidth, in designing such a system, we must address the following questions: 1) What are efficient replication algorithms for video data which do not incur significant system overheads and 2) what are appropriate trigger mechanisms for initiating replication or deletion of a video object. A poor choice of triggers can cause the system to perform unnecessary replication or deletion, which can lead to a waste of system resources, e.g., if a deleted object is replicated in the "near" future.

There are multiple classes of replication policies that can be studied. Briefly, the basic trade-off is as follows: The shorter the replication period, the faster one can achieve a load balanced system which should result in better system performance. On the other hand, more "aggressive" replication policies (which result in shorter replication periods) can interfere with "normal" operation of the system (e.g., by using more resources for replication) and, thus, may result in worse system performance. The topic of "appropriate" replication policies is outside the scope of this paper (refer to [18]). However, efficient solution techniques discussed earlier, as well as their application to VOD systems discussed in this section, should aid designers in efficient experimentation with the different replication policies. In this paper, we concentrate on the second question above, i.e., the question of proper trigger mechanisms. More specifically, we model *threshold-based* trigger methods with *hysteresis* behavior. Since the cost of altering the number of replicas is significant, the use of a threshold-based approach can result in a cost-controlled creation and deletion of these replicas, according to the changes in the access patterns [2]. The "general" benefits of using threshold-based policies with hysteresis are discussed in Section 1.

We model the distributed VOD system with K nodes and threshold-based replication of video objects (i.e., decisions of addition and removal of object copies are threshold-based), as a K-server threshold-based queuing system with hysteresis. This model is defined as follows: Each copy of an object is modeled as a server in this queuing system; thus, in the remainder of the paper, the two terms are considered equivalent. Note that, since a *single* copy of an object residing on one of K nodes of the VOD server is able to "serve" *multiple*, let us say a maximum of C, requests for the same video, each of the K servers in the queuing system is able to serve up to C simultaneous requests and will be modeled as an $M/M/C$ server [16].

Moreover, replication and deletion of objects is equivalent to addition and removal of servers. Clearly, the creation of a new copy of an object is not an instantaneous operation, hence the need for the noninstantaneous activation of servers. However, it is fair to consider deletion of a replica as an instantaneous operation since all we basically have to do is mark the object as deleted (depending on the details of the algorithms used, some amount of time might be required to delete a copy; however, this would be a relatively small amount of time, i.e., relative to the amount of time it takes to create a new copy).

Fig. 8 illustrates our model of the VOD system as a threshold-based queuing system with hysteresis and noninstantaneous server activation with the following parameters: 1) α is the time needed to activate a server (i.e., create a copy) and is a function of the replication policy used, as well as the amount of server resources available for replication, 2) δ is the number of streams currently being served by all active servers when some of the servers have

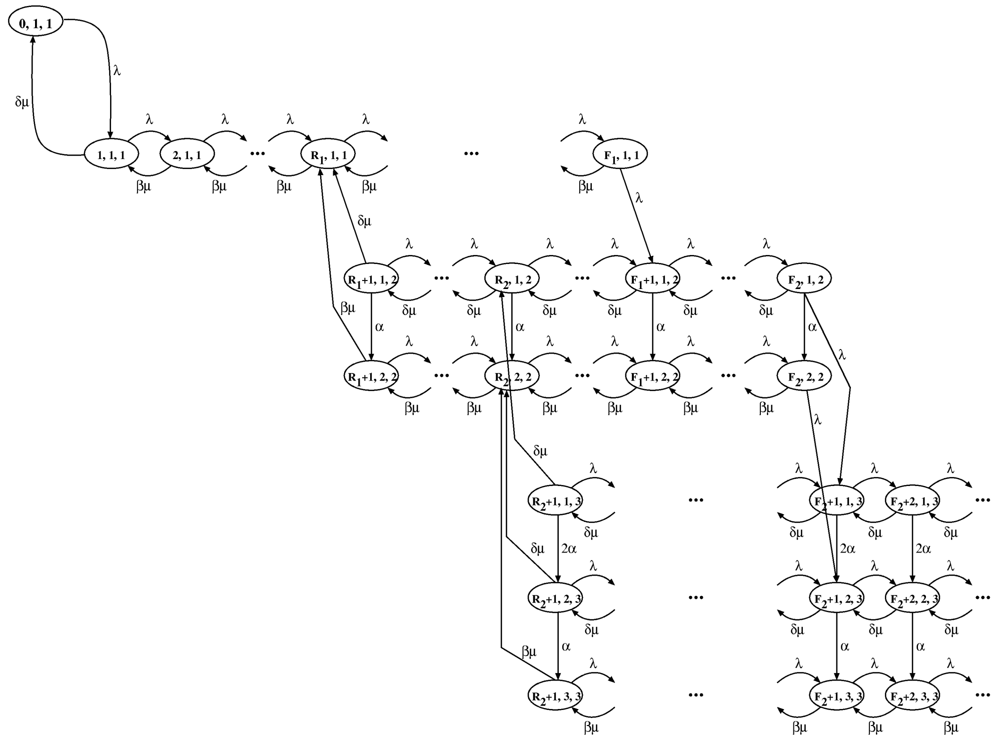


Fig. 8. Application of model to VOD system.

not been activated yet, as prescribed by the threshold vectors, and is a function of the current user population as well as the replication policy, and 3) β is the number of streams currently being served by all active servers when all servers have been activated, as prescribed by the threshold vectors, and is a function of the current user population. In general, α, δ, and β are all functions of the current state (i.e., (i, j, l)) and the specific values of α and δ depend on the replication policy being used. For instance, in the case of a replication policy which requires one stream worth of server resources to accomplish the replication (refer to [18] for details), δ and β can be expressed as follows: $\delta = min(i, j * (C - 1))$ and $\beta = min(i, j * C)$, where C is the maximum number of streams that one server can serve simultaneously. These minor modifications to the model parameters, as compared to the ones depicted in Fig. 1, are needed because 1) a single copy of an object is able to service up to C simultaneous video streams and 2) some of the servers' resources are potentially being used for replication purposes.

In summary, we can use the methodology described in the previous sections to analyze candidate designs (e.g., various threshold settings) and the subsequent performance of a distributed VOD system. Next, we illustrate such an application of our methodology to a VOD system through numerical examples.

8 Numerical Examples and Validation of Bounds

In this section, we present numerical examples which empirically illustrate the computational savings as well as the tightness of the bounds given in Sections 5 and 6 (*theoretical* results are given in Section 4.1).

We define the computational savings as the number of flops used to solve the original model $\mathcal{M}$ (as described in Section 2) divided by the number of flops used to solve the modified model(s) (as described in Sections 5 and 6), i.e., computational savings which are greater than 1 are a reduction in computational cost.

We begin with the original model of a threshold-based queuing system (refer to Section 2) with hysteresis (i.e., without considering the extensions given in Section 7). We have three cases where the number of servers, K, is equal to 5, 7, and 10, respectively. For $K = 5$, the threshold vectors are $\boldsymbol{F} = (25, 50, 75, 100)$ and $\boldsymbol{R} = (12, 24, 49, 74)$, for $K = 7$, the threshold vectors are $\boldsymbol{F} = (25, 50, 75, 100, 125, 150)$ and $\boldsymbol{R} = (12, 24, 49, 74, 85, 110)$ and, for $K = 10$, the threshold vectors are $\boldsymbol{F} = (25, 50, 75, 100, 125, 150, 175, 200, 225)$ and $\boldsymbol{R} = (12, 24, 49, 74, 85, 110, 130, 160, 190)$. For all values of K, C is set to 30 and $\alpha = 1.0$. The service rate μ is equal to 1.

TABLE 1
Illustration of Computational Savings with Different Values of K

K	λ	System utilization	Computational Saving
5	15.0	0.10	8.457
5	145.0	0.96	8.331
7	21.0	0.10	13.264
7	203.0	0.96	13.174
10	30.0	0.10	28.500
10	290.0	0.96	28.370

TABLE 2
Tightness of Bounds for $\alpha = 20\mu$

λ	Utilization	$\mathcal{M}_1^u$ expected response time	$\mathcal{M}_2^u$ expected response time	$\mathcal{M}_1^l$ expected response time	$\mathcal{M}_2^l$ expected response time	% Error
30.0	0.1	0.033333	0.033333	0.033333	0.033333	——
60.0	0.2	0.169967	0.169967	0.169967	0.169967	——
90.0	0.3	0.216872	0.216872	0.216872	0.216872	——
120.0	0.4	0.244564	0.244550	0.244517	0.244517	0.013311
150.0	0.5	0.254905	0.254916	0.254809	0.254809	0.037531
180.0	0.6	0.278554	0.278533	0.278424	0.278424	0.039070
210.0	0.7	0.299463	0.299451	0.299287	0.299287	0.054792
240.0	0.8	0.313259	0.313247	0.313081	0.313081	0.052827
270.0	0.9	0.328654	0.328680	0.328450	0.328450	0.062080
290.0	0.96	0.383838	0.384005	0.383705	0.383705	0.034608

TABLE 3
Tightness of Bounds for $\alpha = 10\mu$

λ	Utilization	$\mathcal{M}_1^u$ expected response time	$\mathcal{M}_2^u$ expected response time	$\mathcal{M}_1^l$ expected response time	$\mathcal{M}_2^l$ expected response time	% Error
30.0	0.1	0.033333	0.033333	0.033333	0.033333	——
60.0	0.2	0.170016	0.170016	0.170016	0.170016	——
90.0	0.3	0.217060	0.217060	0.217060	0.217060	——
120.0	0.4	0.244746	0.244718	0.244653	0.244653	0.026672
150.0	0.5	0.255113	0.255136	0.254922	0.254922	0.074931
180.0	0.6	0.278765	0.278722	0.278504	0.278504	0.078124
210.0	0.7	0.299720	0.299697	0.299369	0.299369	0.109608
240.0	0.8	0.313505	0.313481	0.313150	0.313150	0.105630
270.0	0.9	0.328935	0.328989	0.328528	0.328528	0.123905
290.0	0.96	0.384070	0.384406	0.383805	0.383805	0.069003

TABLE 4
Tightness of Bounds for $\alpha = \mu$

λ	Utilization	$\mathcal{M}_1^u$ expected response time	$\mathcal{M}_2^u$ expected response time	$\mathcal{M}_1^l$ expected response time	$\mathcal{M}_2^l$ expected response time	% Error
30.0	0.1	0.033333	0.033333	0.033333	0.033333	——
60.0	0.2	0.170891	0.170891	0.170891	0.170891	——
90.0	0.3	0.220488	0.220489	0.220485	0.220485	0.001309
120.0	0.4	0.248024	0.247733	0.247051	0.247051	0.276299
150.0	0.5	0.258797	0.259123	0.256944	0.256944	0.721062
180.0	0.6	0.282530	0.282085	0.279912	0.279912	0.776326
210.0	0.7	0.304202	0.304118	0.300824	0.300824	1.095207
240.0	0.8	0.317900	0.317658	0.314356	0.314356	1.050094
270.0	0.9	0.333847	0.334613	0.329931	0.329931	1.187057
290.0	0.96	0.388147	0.391621	0.385635	0.385635	0.651192

We vary the arrival rate so that we have either a lightly loaded system (with system utilization of 0.1) or a highly loaded system (with system utilization of 0.96). Table 1 depicts the computational saving. Note that the solution of all models (i.e., upper and lower bound models and the original model) is carried out using the MATLAB numerical solutions package. From this table, it is clear that we have large computational savings which grow as the model grows, e.g., either as the number of servers, K, increases and/or as the differences between the forward and reverse thresholds, F_i and R_i, grow.

For the original model defined in Section 2, Tables 2, 3, and 4 illustrate the tightness of the bounds for the expected response time metric under the following settings: $K = 5$, $\boldsymbol{F} = (25, 50, 75, 100)$, and $\boldsymbol{R} = (12, 24, 49, 74)$. The service rate μ is set to 60, the average arrival rate λ is varied from 30 to 290, while α is equal to 20μ, 10μ, and μ, respectively. In all cases, the percentage of error (%E) is defined to be:

$$\text{percentage error} = \frac{\text{spread of the bounds}}{\text{lower bound}} \times 100\%, \tag{30}$$

where: 1) the "spread of the bounds" is the difference between the minimum of the two upper bounds and the maximum of the two lower bounds and 2) "lower bound" refers to the maximum of the two lower bounds. As can be seen from these tables, the obtained bounds are tight. The percentage error is less than 1.2 percent for all illustrated cases, even under high system utilization.

In Fig. 9, we depict the expected response time of the system under various threshold vectors. Since the percentage error is very small, we simply plot the expected response time given by the tighter of the two upper bounds. As can be seen from this figure, significant differences in the expected response time are produced by different choices of the system threshold settings. This occurs under low, moderate, and high system utilizations. Thus, our efficient solution methodology for threshold-based models can be of

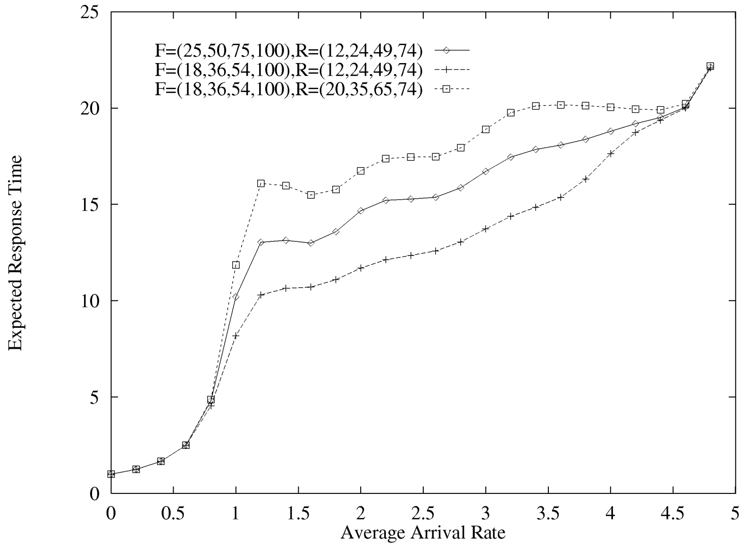


Fig. 9. Expected response time under different threshold vectors, $K = 5$ and $\mu = 1.0$.

great use in computational-cost-efficient experimentation with system parameter settings.

In the remainder of this section, we give numerical results for the model corresponding to the VOD applications. We set $C = 30$, $\delta = \beta = min(i, j * C)$, and $K = 7$. In Tables 5, 6, 7, and 8, we depict the tightness of the bounds under different values of the forward threshold vector ($\boldsymbol{F}$), the reverse threshold vector ($\boldsymbol{R}$), and the rate of server activation (α). We note that we experimented with other values of K, $\boldsymbol{F}, \boldsymbol{R}$, and α, and the results were qualitatively similar; thus, we do not present them here. (Also note that "upper bound" refers to the minimum of the two upper bounds and "lower bound" refers to the maximum of the two lower bounds.)

As can be seen from these tables, the obtained bounds are tight for various values of $\boldsymbol{F}$, $\boldsymbol{R}$, and α. The percentage error is less than 0.8 percent for all illustrated cases. This implies that, even when server activation rate α is relatively small, as compared to the service rate μ (e.g., $\alpha = 0.01$ and $\mu = 1.0$), we can still obtain tight performance bounds with significantly reduced computational costs. It is also important to point out that the percentage error is small, even under *high* system utilizations. This is due to the fact that, in the heavily loaded cases, the system spends most of its time operating in the unmodified region (or the tail end) of the state space. Furthermore, note that a system with threshold settings that correspond to less aggressive replication but more aggressive dereplication of video objects (i.e., Table 6) can achieve comparable performance (in terms of expected response time) to a system with more aggressive replication of video objects (i.e., Tables 5 and 7). Systems with more aggressive replication policies tend to have a higher operational cost (i.e., there is a cost associated with operating a greater number of servers, hence the original motivation for use of threshold-based policies). Thus, by properly setting the threshold vectors, we can obtain lower system operating costs without a loss in system performance.

In summary, an important contribution of our methodology is that it facilitates *efficient* (in terms of computational cost) experimentation with possible system designs (e.g., as in the case of the VOD server application) for systems using threshold-based resource management policies with hysteresis behavior.

TABLE 5
Tightness of Bounds for $K = 7$, $\alpha = 1.0$, $C = 30$, $F = [25, 50, 75, 100, 125, 150]$, $R = [12, 24, 49, 74, 85, 110]$

λ	Utilization	upper bound expected response time	lower bound expected response time	% Error
21.0	0.1	1.000000255862361e+00	1.000000255862172e+00	1.891819549916208e-11
42.0	0.2	1.000000309530742e+00	1.000000000075587e+00	3.094551548109907e-05
63.0	0.3	1.000334567282336e+00	1.000000000000281e+00	3.345672820548488e-02
84.0	0.4	1.000593861233585e+00	1.000000000000034e+00	5.938612335509650e-02
105.0	0.5	1.000288177154562e+00	1.000000000000001e+00	2.881771545608644e-02
126.0	0.6	1.000312596911268e+00	1.000000000000063e+00	3.125969112049790e-02
147.0	0.7	1.000041019171975e+00	1.000000009869768e+00	4.100930180243217e-03
168.0	0.8	1.000025498901918e+00	1.000025385141033e+00	1.137579971716200e-05
189.0	0.9	1.004170068823032e+00	1.004170068777971e+00	4.487371745694744e-09
203.0	0.96	1.074135437109156e+00	1.074135437109065e+00	8.475494325397538e-12

TABLE 6
Tightness of Bounds for $K = 7,\ \alpha = 1.0,\ C = 30,\ F = [30, 60, 90, 120, 150, 180],\ R = [20, 50, 80, 110, 140, 170]$

λ	Utilization	upper bound expected response time	lower bound expected response time	% Error
21.0	0.1	1.000104623069284e+00	1.000104623068440e+00	8.439032515723980e-11
42.0	0.2	1.000343607715703e+00	1.000009245892043e+00	3.343587322152371e-02
63.0	0.3	1.001834121837977e+00	1.000049244341178e+00	1.784789606010720e-01
84.0	0.4	1.004670331219891e+00	1.000045861652261e+00	4.624257491541032e-01
105.0	0.5	1.003747332082535e+00	1.000023048231660e+00	3.724198014696275e-01
126.0	0.6	1.004389788045049e+00	1.000023817269108e+00	4.365866792916622e-01
147.0	0.7	1.006136578130167e+00	1.000023759082076e+00	6.112673816572111e-01
168.0	0.8	1.006148865501521e+00	1.000043826844377e+00	6.104771104290813e-01
189.0	0.9	1.007465564351478e+00	1.004180690086566e+00	3.271198398197550e-01
203.0	0.96	1.074796618224905e+00	1.074137906388060e+00	6.132469889828774e-02

TABLE 7
Tightness of Bounds for $K = 7,\ \alpha = 1.0,\ C = 30,\ F = [20, 40, 60, 80, 100, 120],\ R = [10, 25, 45, 50, 90, 110]$

λ	Utilization	upper bound expected response time	lower bound expected response time	% Error
21.0	0.1	1.000000001708815e+00	1.000000000741306e+00	9.675091831628907e-08
42.0	0.2	1.000117054064421e+00	1.000000000000175e+00	1.170540642460933e-02
63.0	0.3	1.000348503764478e+00	1.000000000000098e+00	3.485037643799696e-02
84.0	0.4	1.001141127258548e+00	1.000000000000045e+00	1.141127258502854e-01
105.0	0.5	1.003067276012174e+00	1.000000000000003e+00	3.067276012170818e-01
126.0	0.6	1.001607716264445e+00	1.000000000000062e+00	1.607716264382953e-01
147.0	0.7	1.000046759779983e+00	1.000000009869766e+00	4.674990975561479e-03
168.0	0.8	1.000025501396460e+00	1.000025385141034e+00	1.162524750891850e-05
189.0	0.9	1.004170068823262e+00	1.004170068777979e+00	4.509483996580184e-09
203.0	0.96	1.074135437109158e+00	1.074135437109065e+00	8.661541761808702e-12

TABLE 8
Tightness of Bounds with Vary α where $K = 7,\ C = 30,\ F = [25, 50, 75, 100, 125, 150],\ R = [12, 24, 49, 74, 85, 110]$

λ	α	Utilization	upper bound expected response time	lower bound expected response time	% Error
21.0	0.01	0.1	1.001227682210043e+00	1.001227478821129e+00	2.031395644199525e-05
21.0	0.10	0.1	1.000072717244459e+00	1.000072716973847e+00	2.705923006178263e-08
21.0	0.20	0.1	1.000019436882976e+00	1.000019436839250e+00	4.372506190325745e-09
21.0	1.00	0.1	1.000000255862361e+00	1.000000255862172e+00	1.891819549916208e-11
21.0	5.00	0.1	1.000000000349613e+00	1.000000000349613e+00	0.000000000000000e+00
21.0	10.00	0.1	1.000000000010455e+00	1.000000000010455e+00	0.000000000000000e+00
203.0	0.01	0.96	1.074135437390643e+00	1.074135437111578e+00	2.598042888207738e-08
203.0	0.10	0.96	1.074135437111311e+00	1.074135437110102e+00	1.125586990286464e-10
203.0	0.20	0.96	1.074135437110465e+00	1.074135437109956e+00	4.738008047267083e-11
203.0	1.00	0.96	1.074135437109156e+00	1.074135437109065e+00	8.475494325397538e-12
203.0	5.00	0.96	1.074135437107669e+00	1.074135437107651e+00	1.674426927702693e-12
203.0	10.00	0.96	1.074135437107182e+00	1.074135437107173e+00	8.475494325412467e-13

9 Conclusions

We have considered a K-server threshold-based queuing system with hysteresis in which the number of servers, employed for serving customers, is governed by forward and reverse threshold vectors. The main motivation for using a threshold-based approach was that many applications incur significant server setup, usage, and removal costs. The motivation for the use of hysteresis was to control the cost during momentary fluctuations in workload. An important and distinguishing characteristic of our work is that we considered the *time to add a server to be nonnegligible*, which is a more accurate model for many applications.

In this work, we have shown that an exact solution of the model can be obtained but at fairly significant computational costs. We then developed an efficient method for computing the steady state probabilities of a multiserver threshold-based queuing system with hysteresis, which, in turn, allowed computation of various performance measures. More specifically, we proposed modified models, which we showed to have an efficient computational solution, and used them to bound the performance measures of interest for the original model. These bounds are tight and the reduction in computational cost, as compared to the exact solution, is significant. We have also illustrated how we can apply this methodology to a VOD system. The example cases presented in the paper resulted in less than a 1.2 percent error (due to bounding) with an order of magnitude reduction in computational cost. The reduction in computational cost should be even greater for larger systems (for instance, we expect to get three orders of magnitude reduction in computational cost when the size of the system grows to approximately 40 servers). It is useful to point out that our bounding methodology can be applied to a heterogenous server system if we include an additional constraint which specifies server activation and deactivation order, e.g., from the slowest to the fastest server during activation and from the fastest to the slowest server during deactivation. An interesting open problem might be to find an ordering of server activation and deactivation that would optimize system operational cost under given performance constraints.

In conclusion, an important contribution of our methodology is that it facilitates *efficient* experimentation with possible system designs for systems which use threshold-based resource management policies with hysteresis behavior. Such experimentation is either inaccurate or infeasible (for reasonably large systems) with previously existing solution methodologies.

ACKNOWLEDGMENTS

This research was supported in part by US National Science Foundation grants CCR-98-96232 and ANI-00-70016 (L. Goulchik) and the RGC Earmarked Grant and the CUHK Direct Grant (J.C.S. Lui).

REFERENCES

[1] S. Berson, S. Ghandeharizadeh, R.R. Muntz, and X. Ju, "Staggered Striping in Multimedia Information Systems," *Proc. ACM SIGMOD,* 1994.
[2] C. Chou, L. Golubchik, and J.C.S. Lui, "Striping Doesn't Scale: How to Achieve Scalability for Continuous Media Servers with Replication," *Proc. 20th Int'l Conf. Distributed Computing Systems,* Apr. 2000.
[3] P.J. Courtois, *Decomposability—Queueing and Computer System Applications.* New York: Academic Press, 1977.
[4] ? Bolosky et al., "The Tiger Video Fileserve," Technical Report MSR-TR-96-09, Michrosoft Research, 1996.
[5] S. Ghandeharizadeh and R.R. Muntz, "Design and Implementation of Scalable Continuous Media Servers," *Parallel Computing J.,* Jan. 1998.
[6] L. Golubchik and J.C.S. Lui, "Bounding of Performance Measures for a Threshold-Based Queuing System with Hysteresis," *Proc. 1997 ACM SIGMETRICS Conf.,* June 1997.
[7] W. Grassman, "Transient Solutions of Markovian Queuing Systems," *Computer & Operations Research,* vol. 4, pp. 47-53, 1977.
[8] S.C. Graves and J. Keilson, "The Compensation Method Applied to a One-Product Production/Inventory Problem," *J. Math. Operational Research,* vol. 6, pp. 246-262, 1981.
[9] R.L. Haskin, "Tiger Shark: A Scalable File System for Multimedia," technical report, IBM Research, 1996.
[10] O.C. Ibe, "An Approximate Analysis of a Multi-Server Queuing System with a Fixed Order of Access," Technical Report RC9346, IBM Research, 1982.
[11] O.C. Ibe and J. Keilson, "Multi-Server Threshold Queues with Hysteresis," *Performance Evaluation,* vol. 21, pp. 185-212, 1995.
[12] O.C. Ibe and K. Maruyama, "An Approximation Method for a Class of Queuing Systems," *Performance Evaluation,* vol. 5, pp. 15-27, 1985.
[13] J. Keilson, *Green's Function Methods in Probability Theory.* London: Charles Griffin, 1965.
[14] J. Keilson, *Markov Chain Models: Rarity and Exponentiality.* New York: Springer, 1979.
[15] P.J.B. King, *Computer and Communication Systems Performance Modeling.* New York: Prentice Hall, 1990.
[16] L. Kleinrock, *Queueing Systems,* vol. 1. Wiley-Interscience, 1975.
[17] R.L. Larsen and A.K. Agrawala, "Control of a Heterogeneous Two-Server Exponential Queuing System," *IEEE Trans. Software Eng.,* vol. 9, pp. 552-526, 1983.
[18] P.W.K. Lie, J.C.S. Lui, and L. Golubchik, "Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems," *Proc. Eighth Int'l Workshop Research Issues in Data Eng.: Continuous-Media Databases and Applications (RIDE '98),* Feb. 1998.
[19] P.W.K. Lie, J.C.S. Lui, and L. Golubchik, "Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems," *J. Multimedia Tools and Applications,* vol. 11, no. 1, May 2000.
[20] W. Lin and P.R. Kumar, "Optimal Control of a Queuing System with Two Heterogeneous Servers," *IEEE Trans. Automatic Control,* vol. 29, pp. 696-703, 1984.
[21] T. Lindvall, *Lectures on the Coupling Method.* Wiley Interscience, 1992.
[22] J.D.C. Little, "A Proof of the Queueing Formula $L = \lambda W$," *Operations Research,* vol. 9, pp. 383-387, May 1961.
[23] J.C.S. Lui and L. Golubchik, "Stochastic Complement Analysis of Multi-Server Threshold Queues with Hysteresis," *Performance Evaluation J.,* vol. 35, nos. 1-2, pp. 19-48, 1999.
[24] J.C.S. Lui, R.R. Muntz, and D. Towsley, "Bounding the Mean Response Time of a Minimum Expected Delay Routing System: An Algorithmic Approach," *IEEE Trans. Computers,* vol. 44, no. 5, pp. 1371-1382, 1995.
[25] A.W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications.* Baltimore: Academic Press, 1979.
[26] C.D. Meyer, "Stochastic Complementation, Uncoupling Markov Chains and the Theory of Nearly Reducible Systems," *SIAM Rev.,* vol. 31, no. 2, pp. 240-272, 1989.
[27] J.A. Morrison, "Two-Server Queue with One Server Idle below a Threshold," *Queueing Systems,* vol. 7, pp. 325-336, 1990.
[28] R.R. Muntz, E. de Souza e Silva, and A. Goyal, "Bounding Availability of Repairable Computer Systems," *Proc. PERFORMANCE '89 and 1989 ACM SIGMETRICS Conf.,* May 1989.
[29] R. Nelson and D. Towsley, "Approximating the Mean Time in System in a Multiple-Server Queue that Uses Threshold Scheduling," *Operations Research,* vol. 35, pp. 419-427, 1987.
[30] M.F. Neuts, *Matrix-Geometric Solutions in Stochastic Models—An Algorithmic Approach.* Baltimore: John Hopkins Univ. Press, 1981.
[31] J.R. Santos and R.R. Muntz, "Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configurations," *Proc. Sixth ACM Multimedia Conf.,* Sept. 1998.
[32] W.J. Stewart, *Introduction to Numerical Solution of Markov Chains.* Princeton Univ. Press, 1994.
[33] D. Towsley, *Application of Majorization to Control Problems in Queueing Systems, in Scheduling Theory and Its Applications,* P. Chretienne, E.G. Coffman Jr., J.K. Lenstra, and Z. Liu, eds. Chichester, U.K.: Wiley, 1995.
[34] J. Wolf, H. Shachnai, and P. Yu, "DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer Systems," *Proc. ACM SIGMETRICS and Performance,* May 1995.

Leana Golubchik received the PhD degree from the Computer Science Department at the University of California at Los Angeles in 1995. She is an associate professor in the Department of Computer Science at the University of Maryland at College Park. From Fall 1995 until Summer 1997, she was an assistant professor in the Department of Computer Science at Columbia University. Her research interests include computer systems modeling and performance evaluation, Internet-based computing, and multimedia storage systems. She is currently the vice chair of SIGMETRICS (elected 2001) and a past member of its Board of Directors (1999-2001). She was a guest coeditor for special issues of the *IEEE Transactions on Knowledge and Data Engineering*, *Parallel Computing*, and the *International Journal of Intelligent Systems*, a program cochair of the 2001 Joint ACM SIGMETRICS/Performance Conference and MIS '99, as well as a program committee member of several conferences, including SIGMETRICS, SIGMOD, ICDCS, ICDE, and PDIS. She has received several awards, including the US National Science Foundation (NSF) CAREER award, the IBM Doctoral Fellowship, and the NSF Doctoral Fellowship. She is a member of the IFIP WG 7.3 (elected 2000), ACM, IEEE Computer Society, and Tau Beta Pi.

John C.S. Lui received the PhD degree in computer science from the University of California at Los Angeles. While he was a graduate student, he participated in a parallel database project in the IBM T.J. Watson Research Center. After his graduation, he joined a team at the IBM Almaden Research Laboratory/San Jose Laboratory and participated in research and development of a parallel I/O architecture and file system project. He later joined the Department of Computer Science and Engineering of the Chinese University of Hong Kong. His current research interests are in communication networks, distributed multimedia systems, OS design issues, parallel I/O, storage architectures, and performance evaluation theory. He is a member of Tau Beta Pi, the ACM, and the IEEE Computer Society. His personal interests include general reading and films.