Bounding of Performance Measures for a Threshold-based Queueing System with Hysteresis

Leana Golubchik
Columbia University
Department of Computer Science
email: leana@cs.columbia.edu

John C.S. Lui
The Chinese University of Hong Kong
Department of Computer Science & Engineering
email: cslui@cs.cuhk.edu.hk

Abstract

In this paper, we consider a K-server threshold-based queueing system with hysteresis in which the number of servers, employed for servicing 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 $R_1 < R_2 < \cdots < R_{K-1}$). There are many applications where a threshold-based queueing 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. And, 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 non-negligible*. This is a more accurate model, for many applications, than previously considered in other works. Our main goal in this work is to develop an efficient method for computing the steady state probabilities of a multi-server threshold queueing system with hysteresis, which will, in turn, allow computation of various performance measures.

1 Introduction

In this paper, we consider a K-server threshold-based queueing system with hysteresis in which the number of servers, employed for servicing 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 multi-server queueing system behaves as follows. A customer arriving to an empty system is serviced by a single server. A new arrival to a system with F_i customers already there forces a *non-instantaneous* activation of one additional server. A departure from a system which leaves R_i customers behind forces a removal of one server.

SIGMETRICS '97 Seattle, WA, USA

The main motivation for using a threshold-based approach is that many systems incur significant server setup, usage, and removal costs. More specifically, under light loads, it is not desirable to operate unnecessarily 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 queueing system can be of great use, e.g., in transport protocols of communication networks [10], 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 service 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, the use of a threshold-based approach can result in a cost-controlled creation and deletion of these connection, according to

the changes in the workload.

As in the case of electronic circuits that are prone to oscillation effects, a "simple" threshold 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 non-negligible 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 towards a heavily loaded operation region, and it is desirable to remove servers only when a system is moving towards 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 queueing systems with hysteresis.

As already mentioned, a threshold-based queueing system with hysteresis is defined by the forward and reverse threshold vectors (see Section 2 for details). Our main goal in this work is to develop an efficient method for computing the steady state probabilities of a multi-server threshold-based queueing system with hysteresis, which will, in turn, allow computation of various performance measures. Thus, the question of optimal or "good" values for the threshold vectors, although a very interesting one, is outside the scope of this paper and is the topic of future work. We must point out, however, that our solution method, due to its intuitive nature, should facilitate accessible experimentation techniques for investigating the "goodness" of various threshold values.

Given the above motivation for the use of threshold-based queueing systems with hysteresis, in this paper we present an efficient technique for computing tight performance bounds for the corresponding Markov chain model. We begin with a very brief survey of the published literature on the threshold-based queueing problem. A two-server system is considered in [12], [13], and [18]. An approximate solution for solving a degenerate form of this problem (where all thresholds are set to zero) is presented in [5, 7]; an approximate solution for a system that employs (non-zero) thresholds is presented in [20] (but without hysteresis). In [6], the authors solve a limited form of the multi-server threshold queueing system with hysteresis, using the Green's function method [4, 8, 9] — 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 2 heterogeneous servers. In [16] exact solutions, using stochastic complementation [17], for the K-server homogeneous, heterogeneous, and bulk arrival variations of the multi-server threshold queueing system with hysteresis are given; no restrictions are placed on the number of servers or the bulk sizes or the size of the waiting room. Stochastic complementation is a more intuitive and a more easily extensible method, and is exploited in this work as well.

In this paper, we consider and solve a homogeneous multi-server threshold-based queueing 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 non-negligible, we consider it an important and distinguishing characteristic of our work. We first give an exact solution for computing the steady state probabilities of our model using the matrix geometric method [21]. However, we feel that the exact solution is not efficient. And, thus, the main contribution of this paper is an *efficient* solution of a threshold-based queueing system with hysteresis obtained through a computation of *tight* performance *bounds.* More specifically, we compute the steady state probabilities of the bounding models using a combination of stochastic complementation [17] and the matrix geometric [21] 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.

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. Section 4 describes our general approach to solving the model. 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. The goodness of these bounds and numerical results are discussed in Section 7. Finally, our conclusions are given in Section 8.

2 Model

In this section we describe our model, which is illustrated in Figure 1. We consider a multi-server threshold-based queueing 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

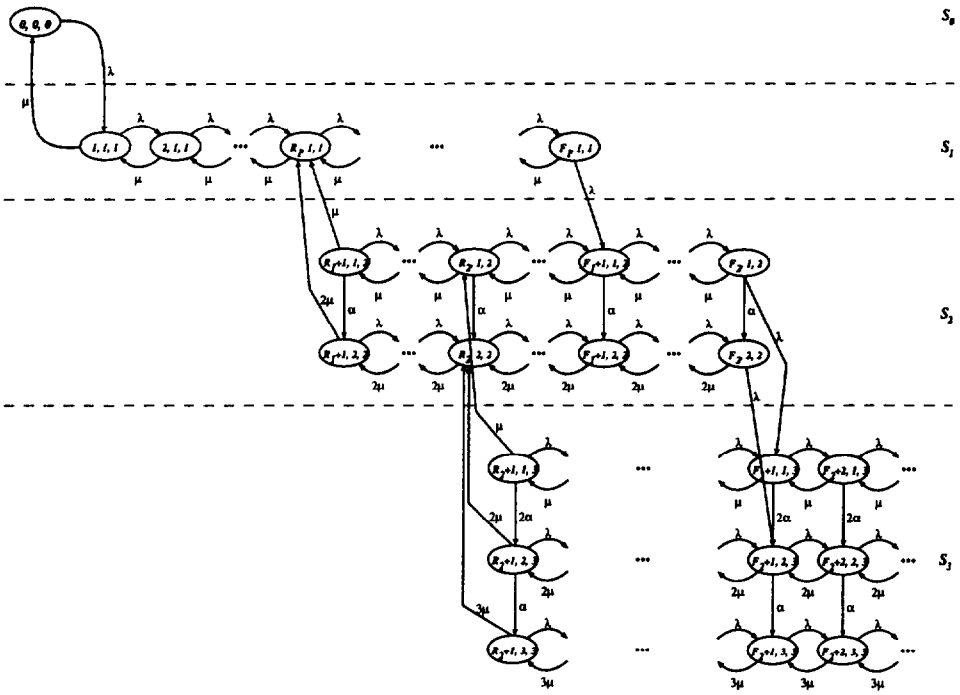

Figure 1: State Transition Diagram for $K = 3$.

this queueing 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[1]. Note that, unlike in [6, 16], the addition of a server is not instantaneous, but is governed by a Poisson process with a rate α (refer to Figure 1). This is motivated by the fact that in many applications addition of a new server takes a non-negligible amount of time. The use of a threshold-based approach can result in a cost-controlled addition and removal of servers.

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

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

where N is the number of customers in the queueing system, N_s is the number of busy servers, and L is the level to which the state belongs — more 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 in our model activation of servers is not instantaneous[2]. Figure 1 illustrates the state transition diagram for the homogeneous servers threshold-based queueing system with hysteresis[3] where $K = 3$. Formally, the transition structure of $\mathcal{M}$ with K homogeneous servers, where K is unrestricted, can be specified as follows:

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

where $\mathbf{1}\{x\}$ is an indicator function, i.e., it is equal to 1 if condition x is true and 0 otherwise.

[1] Note that, there are multiple ways to create a total order between the F_i's and the R_i's; 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.

[2] The "level" part of the state description is somewhat artificial at this point but will become useful, later in the paper, in constructing a solution to this model.

[3] The S_i notation, in this figure, will be defined in Section 4.

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 [21], 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:

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

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 solution for matrix $\mathbf{R}$ can be obtained using the following iterative procedure:

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

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

Let $\boldsymbol{\pi}_i$ be the steady state probability vector for states in the set B_i where $i \geq 0$. We can express $\boldsymbol{\pi}_i$ as:

$$\boldsymbol{\pi}_j = \boldsymbol{\pi}_1\mathbf{R}^{j-1} \quad j = 2, 3, \ldots \quad (5)$$

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

$$\begin{aligned} \boldsymbol{\pi}_0\mathbf{B}_{0,0} + \boldsymbol{\pi}_1\mathbf{B}_{1,0} &= \mathbf{0} \\ \boldsymbol{\pi}_0\mathbf{B}_{0,1} + \boldsymbol{\pi}_1\mathbf{A}_1 + \boldsymbol{\pi}_2\mathbf{A}_2 &= \mathbf{0} \end{aligned} \quad (6)$$

which can be written in matrix form as

$$(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1) \begin{bmatrix} \mathbf{B}_{0,0} & \mathbf{B}_{0,1} \\ \mathbf{B}_{1,0} & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 \end{bmatrix} = \mathbf{0} \quad (7)$$

where we substitute $\boldsymbol{\pi}_2 = \boldsymbol{\pi}_1\mathbf{R}$ in (5) to obtain the submatrix in the lower right-hand corner of (7). To determine the steady state probabilities of all states, we need the normalization constraint, which is:

$$1 = \boldsymbol{\pi}_0 e + \boldsymbol{\pi}_1 \sum_{j=1}^{\infty} \mathbf{R}^{j-1} e = \boldsymbol{\pi}_0 e + \boldsymbol{\pi}_1 (I - \mathbf{R})^{-1} e$$

We can determine the solution of $\boldsymbol{\pi}_j$ by solving the following system of linear equations:

$$(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1) \begin{bmatrix} e & \mathbf{B}^*_{0,0} & \mathbf{B}_{0,1} \\ (I - \mathbf{R})^{-1} e & \mathbf{B}^*_{1,0} & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 \end{bmatrix} = [1, \mathbf{0}] \quad (8)$$

where $\mathbf{M}^*$ is a matrix $\mathbf{M}$ with its first column eliminated.

3.2 Stochastic Complementation

In this section, we briefly review the concept of stochastic complementation [17, 19, 15][4]. 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:

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

and $\boldsymbol{\pi} = [\boldsymbol{\pi}_A, \boldsymbol{\pi}_B]$ is the corresponding steady state probability vector of $\mathcal{M}$. In what follows, we define the notion of stochastic complementation and quote some useful results [17].

Definition 1 *The stochastic complement of $P_{A,A}$, denoted by $C_{A,A}$, is:*

$$C_{A,A} = P_{A,A} + P_{A,B}[I - P_{B,B}]^{-1} P_{B,A} \quad (9)$$

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 $\boldsymbol{\pi}_{|A}$ be the stationary state probability vector for the stochastic complement $C_{A,A}$, then*

$$\boldsymbol{\pi}_{|A} = 1/(\boldsymbol{\pi}_A e)\boldsymbol{\pi}_A \quad (10)$$

where e is the column vector with all entries equal to 1.

[4]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 [3].

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

Let diag($\boldsymbol{v}$) be a diagonal matrix where the i^{th} diagonal element is the i^{th} element of the vector $\boldsymbol{v}$. We can re-write Equation (9) as:

$$\boldsymbol{C}_{A,A} = \boldsymbol{P}_{A,A} + \text{diag}(\boldsymbol{P}_{A,B}\boldsymbol{e})\boldsymbol{Z} \quad (11)$$

where

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

and $\boldsymbol{P}^{*}_{A,B}$ is simply $\boldsymbol{P}_{A,B}$ but with all rows normalized to sum to 1. Let r_i be the i^{th} diagonal element of diag($\boldsymbol{P}_{A,B}\boldsymbol{e}$). 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 $\boldsymbol{z}_i$ be the i^{th} row of $\boldsymbol{Z}$; then we can re-write Equation (11) as:

$$\boldsymbol{C}_{A,A} = \boldsymbol{P}_{A,A} + \begin{bmatrix} r_1\boldsymbol{z}_1 \\ r_2\boldsymbol{z}_2 \\ \vdots \\ r_n\boldsymbol{z}_n \end{bmatrix} \quad (12)$$

Remarks: the probabilistic interpretation of Equation (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 $\boldsymbol{P}_{A,A}$ is therefore equal to $\boldsymbol{P}_{A,A}$ plus any transition probabilities, which used to go from A to B, "folded" back to A and redistributed according to the stochastic matrix $\boldsymbol{Z}$. This interpretation implies that the i^{th} row of matrix $\boldsymbol{Z}$ determines how r_i should be redistributed back to A. In general, it is not an easy task to compute $\boldsymbol{Z}$, but for some special cases where sufficient "structure" exists in the original Markov chain, $\boldsymbol{Z}$ 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:*

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

where $\boldsymbol{Q}_{i,j}$ is the transition rate sub-matrix corresponding to transitions from partition i to partition j. If $\boldsymbol{Q}_{B,A}$ has all zero entries except for some non-zero entries in the i-th 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 $\boldsymbol{\pi}_{|A}$ and is the solution of the following system of linear equations:

$$\begin{aligned} \boldsymbol{\pi}_{|A}\left[\boldsymbol{Q}_{A,A} + \boldsymbol{Q}_{A,B}\ \boldsymbol{e}\ \boldsymbol{e}_i^T\right] &= \boldsymbol{0} \\ \boldsymbol{\pi}_{|A}\ \boldsymbol{e} &= 1 \end{aligned}$$

where $\boldsymbol{e}_i^T$ is a row vector with a 0 in each component, except for the i-th component, which has the value of 1.

Proof: This is intuitively clear based on the stochastic complementation argument above. The matrix $\boldsymbol{Z}$ will have identical rows where each row is equal to $\boldsymbol{e}_i$. This is true because no matter how B is entered, A is entered from B via the i^{th} state, with probability 1. ∎

4 General Approach

In this section we describe a general approach to solving the model presented in Section 2.

4.1 Matrix Geometric Solution

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

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

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

then, $\mathcal{M}$ has a matrix geometric solution [21] as described in Section 3.1, where the components of each of the $\mathbf{A}_i$ submatrices are:

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

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

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

Solving a model using the matrix geometric method involves: (a) solving the $\mathbf{R}$ matrix, using the procedure in Equations (3) and (4), for the repetitive part, and (b) 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 Equations (2) and (13). As

long as both matrices are "reasonably small", we could obtain an efficient solution. Solving the $\mathbf{R}$ matrix, i.e., step (a) above, requires a computational cost of $O(K)^3$. 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:

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

Then, for step (b) above, the exact analysis requires: $O(n_0+n_1+n_2+n_3+....+C_K)^3$ 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 Figure 1).

One approach to reducing the computational complexity is to partition $\mathcal{M}$ into sub-parts, solve each subpart separately, and then combine the solutions, i.e., through the method of decomposition [1]. In our model, we can partition $\mathcal{M}$ into K sub-parts, 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[5]. More specifically, the cost for solving all K sub-parts will be $O(n_0)^3 + O(n_1)^3 + O(n_2)^3 + ...+ O(G_K)$, 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:

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

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

and the $\mathbf{A}_i$'s are given in Equations (15)-(17), and then apply the solution of Section 3.1. In this case, the computational cost for solving $\mathbf{R}$ is also $O(K)^3$, 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, using stochastic complementation (see Section 3.2), we could 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 [1], 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). Lastly, we can compute the individual (unconditional) steady state probabilities of the original Markov process $\mathcal{M}$ [1]; these can in turn be used to compute various related 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 Equation (11)).

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: (a) our bounds are very tight and (b) the bounding technique results in significant computational savings are given in Section 7.

4.3 Analysis of the Aggregated Process

In this section we briefly describe the analysis of the aggregated process. 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 Figure 2. The transition rates of the

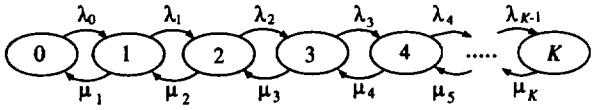

Figure 2: State transition diagram for aggregated process.

aggregated process can be computed as follows:

$$\lambda_0 = \lambda \quad (21)$$

[5]The computational cost of solving the aggregate model is only $O(K)^3$; see Section 4.3 for details.

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

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

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 Figure 2 is a simple birth-death process, and hence the corresponding steady state probabilities are very simple to compute (refer to [11]).

Once we determine, for each l, where $0 \leq l \leq K$,: (1) $\pi_l(i,j)$'s, 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:

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

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 for our system are the expected number of customers and the expected response time.

We can easily compute $N_{\mathcal{S}_l}$, the expected number of customers[6], given that the system is in $\mathcal{S}_l$, for $0 \leq l \leq K-1$, by expressing it as a Markov reward functions, where $R(i,j,l) = i$. Computation of $N_{\mathcal{S}_K}$, the expected number of customers given that the system is in $\mathcal{S}_K$, is a bit more tricky. It is as follows:

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

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

[6] And then, of course, the expected response time using Little's result [14].

5 Upper Bound

In this section, we describe a model which can provide an upper bound on the desired performance measures for the model described in Section 2, namely, the mean number of customers and mean system response time. We begin by illustrating the upper bound idea through an example. Then we present our proof and lastly the computational procedure for obtaining the desired performance measures.

5.1 Upper Bound Model

The intuition for the construction of the upper bound model, $\mathcal{M}^u$, is as follows. We alter several transitions in the original model while satisfying the criteria that the new model will: (1) provide (hopefully a 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 $\mathcal{M}^u$ will have less active servers, that is, $\mathcal{M}^u$ and $\mathcal{M}$ will "see" the same arrivals, but at any given moment, $\mathcal{M}^u$ 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 7).

We begin at the lowest level — for instance, in the case of Figure 1, we begin at level $\mathcal{S}_3$. To achieve the upper bound, we can alter the following transitions. 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 as follows. We can construct a corresponding Markov process, $\mathcal{M}^u$, with the following state space $\mathcal{S}^u$:

$$\mathcal{S}^u = \{(N^u, N_s^u, L^u) \mid$$

$$N^u \geq 0, N_s^u \in \{0, 1, 2, \ldots, K\}, L^u \in \{0, 1, 2, \ldots, K\}\}$$

where N^u is the number of customers in the queueing system, N_s^u is the number of busy servers, and L^u is the level to which the state belongs (see Section 2 for explanation of the "level" notation). The transition structure

of $\mathcal{M}^u$, is the same as that of the original process $\mathcal{M}$, given in Equation (1), except for the transition corresponding to a change of levels due to a departure (line 5 in Equation (1)). The upper bound model transitions that replace this original transition can be specified as follows:

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

5.2 Proof for the upper bound model $\mathcal{M}^u$

In this section, we give a theorem which proves that $\mathcal{M}^u$ 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 busy server 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 $\mathcal{M}^u$ as, $[N^u(t), N_s^u(t), L^u(t)]$.

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

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

Proof: Omitted due to lack of space; please refer to [2]. ∎

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

In this section, we present a numerical computation procedure for the upper bound model $\mathcal{M}^u$. 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) the S_i's are defined as in Equation (18), but with respect to the upper bound model $\mathcal{M}^u$. Then, we can state the following theorem.

Theorem 5 **Multiple entries.** *Given an irreducible Markov process, $\mathcal{M}^u$, of Section 5.1, with state space $\mathcal{S}^u$, 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 sub-matrix corresponding to transitions from partition i to partition j. Then, the r_k's and the z_k's in Equation (12) are as follows:

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

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

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 $\mathcal{M}^u$. Then, the stationary state probability vector, $\boldsymbol{\pi}_{|\mathcal{S}_l^+}$ is given by Equation (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 due to lack of space. 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 $\mathcal{M}^u$. The basic idea is that we first compute the steady state probability vector given that the system $\mathcal{M}^u$ is in a particular set, namely, $\mathcal{S}_l$ for $0 \leq l \leq K$. Computation of the aggregate state probabilities for each $\mathcal{S}_l$ is 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 — the mean number of customers and the mean response time.

Let us now concentrate on computation of the steady state probabilities given that the system $\mathcal{M}^u$ 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 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 [22]). Once that is done, we can easily compute $N_{\mathcal{S}_2}$.

For level $\mathcal{S}_l$, where $3 \leq l \leq K-1$, 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 $1 \leq j \leq l$, 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 states 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 state probabilities given that the system is in $\mathcal{S}_l$ using a variety of methods (refer to [22]). Once the conditional steady state probability vector is determined, we can easily compute $N_{\mathcal{S}_l}$, for $3 \leq l \leq K-1$.

The computation of the conditional 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., the quasi birth-death version of the matrix geometric form, the remainder of the solution can proceed as described in Section 4. Note that, it is possible to have an alternative upper bound model. For details, please refer to [2].

6 Lower Bound

In this section, we describe a model 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 model, $\mathcal{M}^l$, is very similar to that of the upper bound model. We alter several transitions in the original model while satisfying the criteria that the new model will: (1) provide (hopefully a tight) lower 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 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 processes, as compared to the original model, such that $\mathcal{M}^l$ will have more active servers, that is, $\mathcal{M}^l$ and $\mathcal{M}$ will "see" the same arrivals, but at any given moment, $\mathcal{M}^l$ will have the same or *more* servers processing these arrivals. Note that these judicious modifications of departure process will allow us to have a tight lower bound as well as an efficient computational procedure (see Section 7).

We begin at the lowest level — for instance, in the case of Figure 1, we begin at level $\mathcal{S}_3$. To achieve the lower bound, we can alter the following transition. 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 as follows. We can construct a corresponding Markov process, $\mathcal{M}^l$, with the following state space $\mathcal{S}^l$:

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

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

$$\begin{array}{lll} (i,j,l) & \longrightarrow & (i-1, \min(j, l-1), l-1) \\ \text{with rate} & j\mu \mathbf{1}\{ & (i-1 = R_k \in \boldsymbol{R}) \wedge (l = k+1) \\ & & \wedge (l \leq 2) \} \\ (i,j,l) & \longrightarrow & (i-1, l-1, l-1) \\ \text{with rate} & j\mu \mathbf{1}\{ & (i-1 = R_k \in \boldsymbol{R}) \wedge (l = k+1) \\ & & \wedge (l \geq 3) \} \end{array} \quad (27)$$

Note that, it is possible to have an alternative lower bound model. For details, please refer to [2]. The proof that $\mathcal{M}^l$ provides a lower bound on the desired performance measures and the numerical computation of these measures using $\mathcal{M}^l$ are very similar to those given in the context of $\mathcal{M}^u$ (see Sections 5.2 and 5.3). Due to lack of space, we omit both.

7 Numerical Examples and Validation of Bounds

In this section, we present numerical examples which illustrate the tightness of the bounds, given in Sections 5 and 6, as well as the reduction in computational cost[7] due to bounding, as compared to the exact solution technique. In the following examples, the number of servers, K, is equal to 5. The forward and reverse threshold vectors are set to $\boldsymbol{F} = (25, 50, 75, 100)$ and $\boldsymbol{R} = (12, 24, 49, 74)$, respectively. The service rate μ is set to 60, and the the average arrival rate λ is varied from 30 to 290. 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.

Firstly, we demonstrate the savings in computational cost due to obtaining bounds as compared to solving the model exactly. Table 1 depicts performance measures, computed using the original model as well as the bounding models, both, for lightly loaded and heavily loaded systems. It illustrates that significant reduction in computational cost can be obtained using our bounding technique. More specifically, in this example, we maintain tightness of bounds while obtaining more than a 10 fold reduction in computational cost, where cost is measured in flops, the number of floating point operations executed in MATLAB. Clearly, the computational

Model	λ	α	average response time	flops
original	30.0	20μ	0.033333	145481380
$\mathcal{M}^u$	30.0	20μ	0.033333	10739407
$\mathcal{M}^l$	30.0	20μ	0.033333	10739406
original	270.0	20μ	0.328451	145721169
$\mathcal{M}^u$	270.0	20μ	0.328654	11003955
$\mathcal{M}^l$	270.0	20μ	0.328451	11003954

Table 1: Illustration of computational savings.

savings will grow as the model grows, e.g., either as the number of servers, K, grows and/or as the differences between the forward and reverse thresholds, F_i and R_i, grow.

Tables 2, 3 and 4 illustrate the tightness of bounds under different values of α, the rate at which server activation occurs. More specifically, α is equal to 20μ, 10μ, and μ in Tables 2, 3 and 4, respectively. In all cases, the percentage error (%E) is defined as follows:

$$\text{percentage error} = \frac{\text{spread of the bounds}}{\text{lower bound}} \text{x}\, 100\% \quad (28)$$

λ	$\mathcal{M}^u$ expected response time	$\mathcal{M}^l$ expected response time	% Error
30.0	0.033333	0.033333	—
60.0	0.169967	0.169967	—
90.0	0.216872	0.216872	—
120.0	0.244564	0.244517	0.019221
150.0	0.254905	0.254809	0.037675
180.0	0.278554	0.278424	0.046691
210.0	0.299463	0.299287	0.058806
240.0	0.313259	0.313081	0.056854
270.0	0.328654	0.328450	0.062109
290.0	0.383838	0.383705	0.034662

Table 2: Tightness of bounds for $\alpha = 20\mu$.

λ	$\mathcal{M}^u$ expected response time	$\mathcal{M}^l$ expected response time	% Error
30.0	0.033333	0.033333	—
60.0	0.170016	0.170016	—
90.0	0.217060	0.217060	—
120.0	0.244746	0.244653	0.038013
150.0	0.255113	0.254922	0.074924
180.0	0.278765	0.278504	0.093714
210.0	0.299720	0.299369	0.117246
240.0	0.313505	0.313150	0.113364
270.0	0.328935	0.328528	0.123885
290.0	0.384070	0.383805	0.069045

Table 3: Tightness of bounds for $\alpha = 10\mu$

λ	$\mathcal{M}^u$ expected response time	$\mathcal{M}^l$ expected response time	% Error
30.0	0.033333	0.033333	—
60.0	0.170891	0.170891	—
90.0	0.220488	0.220485	0.001360
120.0	0.248024	0.247051	0.393845
150.0	0.258797	0.256944	0.721168
180.0	0.282530	0.279912	0.935293
210.0	0.304202	0.300824	1.122915
240.0	0.317900	0.314356	1.127384
270.0	0.333847	0.329931	1.186914
290.0	0.388147	0.385635	0.651393

Table 4: Tightness of bounds for $\alpha = \mu$

As can be seen from these tables, the obtained bounds are very tight. The percentage error is less than 1.2% for all cases illustrated here, even under high system utilization. It is important to point out that the percentage error is very small at *very high* utilizations, because in the very heavily loaded cases the system spends most of its time operating in the unmodified region (or the tail end) of the state space.

8 Conclusions

In summary, we have considered a K-server threshold-based queueing system with hysteresis in which the num-

[7] More specifically, in this section we present *empirical* evidence of computational savings resulting from bounding; *theoretical* results are given in Section 4.1.

ber of servers, employed for servicing 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 non-negligible*, 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 multi-server threshold-based queueing 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. 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.

Acknowledgement: The authors are grateful to the anonymous referees for their helpful and insightful comments. Leana Golubchik's research was supported in part by the NSF CAREER grant CCR-96-25013. John C.S. Lui's research was supported in part by the RGC and CUHK Direct Grant.

References

[1] P. J. Courtois, "Decomposability : queueing and computer system applications", *ACM monograph series, Academic Press*, New York, 1977.

[2] L. Golubchik and John C.S. Lui, "Bounding of Performance Measures for a Threshold-based Queueing System with Hysteresis", Technical Report, CS-TR-96-10, The Chinese University of Hong Kong.

[3] W.K. Grassman, "Transient Solutions in Markovian Queueing Systems", *Computer and Operation Research*, Vol. 4, pp. 47-53, 1977.

[4] S.C. Graves and J. Keilson, "The Compensation Method Applied to a One-product Production/Inventor Problem", *Journal of Math. Operational Research*, Vol. 6, pp. 246-262, 1981.

[5] O.C. Ibe, "An Approximate Analysis of a Multi-server Queueing System with a Fixed Order of Access". *IBM Research RC9346*, 1982.

[6] O.C. Ibe and J. Keilson, "Multi-server threshold queues with hysteresis", *Journal of Performance Evaluation*, Vol. 21, pp. 185-212, 1995.

[7] O.C. Ibe and K. Maruyama, "An Approximation Method for a Class of Queueing Systems", *Journal of Performance Evaluation*, Vol. 5, pp. 15-27, 1985.

[8] J. Keilson, "Green's Function Methods in Probability Theory", *Charles Griffin*, London, 1965.

[9] J. Keilson, "Markov Chain Models: Rarity and Exponentiality", *Springer*, New York, 1979.

[10] P.J.B. King, "Computer and Communication Systems Performance Modeling", *Prentice-Hall, New York*, 1990.

[11] L. Kleinrock, "Queueing Systems, Volume I", *Wiley-Interscience*, 1975.

[12] R.L. Larsen and A.K. Agrawala, "Control of a heterogeneous two-server exponential queueing system", *IEEE Trans. on Software Engineering*, Vol 9 pp. 552-526, 1983.

[13] W. Lin and P.R. Kumar, "Optimal Control of a Queueing System with two Heterogeneous Servers", *IEEE Trans. on Automatic Control*, Vol. 29, pp. 696-703, 1984.

[14] J. D. C. Little, "A Proof of the Queueing Formula $L = \lambda W$", *Operations Research*, Vol. 9, pp. 383-387, 1961.

[15] John. 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. on Computers*, 44(5), pp. 1371-1382, 1995.

[16] John. C.S. Lui and L. Golubchik, "Stochastic Complement Analysis of Multi-Server Threshold Queues with Hysteresis", Columbia University, CUCS-038-96, 1996.

[17] C.D. Meyer, "Stochastic Complementation, Uncoupling Markov Chains and the Theory of Nearly Reducible Systems", *SIAM Review*, Vol. 31, No. 2, pp. 240-272, 1989.

[18] J.A. Morrison, "Two-server queue with One Server Idle Below a Threshold", *Queueing Systems*, Vol. 7, pp. 325-336, 1990.

[19] R. R. Muntz, E. de Souza e Silva, and A. Goyal, "Bounding Availability of Repairable Computer Systems", *Proceedings of the ACM SIGMETRICS and Performance Conf.*, May 1989.

[20] 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.

[21] M. F. Neuts, "Matrix-geometric Solutions in Stochastic Models - an Algorithmic Approach", *John Hopkins University Press*, 1981.

[22] William J. Stewart, "Introduction to Numerical Solution of Markov Chains", *Princeton University Press*, 1994.

[23] J. Wolf, H. Shachnai and P. Yu, "DASD Dancing A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer Systems", *Proceedings of the ACM SIGMETRICS and Performance Conf.*, May 1995.