Provisioning and Controlling Differentiated Quality of Service in Web Servers: An Analytical Framework

by

Mohammad Mamunur Rashid

A thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the degree of

MASTER OF SCIENCE

Department of Electrical & Computer Engineering

University of Manitoba

Winnipeg, Manitoba, Canada

Provisioning and Controlling Differentiated Quality of Service in Web Servers: An Analytical Framework

BY

Mohammad Mamunur Rashid

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

To My Family...

Abstract

Provisioning quality of service (QoS) in web servers has gained immense importance. To ensure pledged QoS, web service providers need control over the allocation of the resources in their web servers. Control is also necessary for reaching the optimal resource allocation through proper service differentiation. In this thesis, we propose and investigate an analytic approach that enables the service providers to deploy a differentiated service policy that offers this control. The proposed service policy is configurable by tunable control parameters. We devise the relationships between the performance measures and these parameters by adopting a unique queueing theoretic approach. Once these relationships are established, we describe how these parameters can be set to their appropriate and optimal values depending on the objectives of the service providers. We illustrate the usefulness of our approach by conducting the analysis on a real web trace.

Acknowledgements

Many thanks go to all those who were instrumental in the development of this thesis. First, I must thank my advisors, Professor Muthucumaru Maheswaran and Professor Ekram Hossain, for their great guidance through my academic studies and this research. Second, I would like to thank Professor Attahiru Sule Alfa for his generous help in developing the models in the thesis. I am also very thankful to Professor Jelena Mišic for serving in my examining committee.

I would like to acknowledge with gratitude the support from TRLabs, Winnipeg, during the course of my graduate studies at University of Manitoba. I also thank the Faculty of Graduate studies for supporting me with the University of Manitoba Graduate Fellowship and a number of other scholarships.

Great appreciation goes to my family, for their unconditional love and always being behind me. I am profoundly indebted to my wife for her continued encouragement and support to successfully accomplish this endeavor.

Table of Content

List of Figures

1 Introduction

Amidst growing competition web service providers are increasingly becoming interested to better serve their clients. The traditional approach of a best-effort single class service policy in web servers are proving inadequate. There are several reasons. First, the diverse set of competing services provided by web servers has different performance requirements. Second, the requests for the web contents compete for the limited resources in the web servers to get serviced. Moreover, in shared hosting facilities, different business and corporate organizations pay for hosting their web contents and expect performances according to the amount they pay to arrange services for their clients. The client satisfaction or expectations are often abstracted in the QoS they receive and thus, the web server has to be QoS aware. Such a QoS-aware web server adopts differentiated service policy to make sure that the clients are serviced according to their expectations and the importance of the service requests. The service providers are often willing to guarantee that certain services or certain clients get a predefined amount of performance from the server, while maintaining a best-effort approach for the rest. The differentiated service policy has to be designed to meet this requirement. In addition to meeting the guarantees, the service providers often look for reaching certain optimization objectives. One example of such objectives could be to provide the optimum performance for the best-effort requests, while maintaining the performance guarantees for QoS requests. Pricing policies may also shape the optimization objectives, which can vary largely according to what the service providers are looking for to achieve.

One simple and straight forward approach for deploying differentiated service is to adopt priority queueing schemes in the web server. A number of research works [AlMa98][EgHe99][BhFr99] have demonstrated that prioritized scheduling of requests in web servers according to their service requirements can give performance benefits to some premium (high priority) class of requests over some non-premium (low priority) requests. The advantage of priority based schemes is that they are simple to implement and offers ease of integration into the existing web servers. However, the strict priority based schemes fail to offer the desired service differentiation when the service providers need to control the allocation of web server resources to meet QoS guarantees. It offers the differentiation without the means to control the differentiation, whereas the service providers need some "control knob" in their hands. Many admission control and scheduling algorithms [KaKn02], [ChMo02], [AbSh02], [LeLu04], [PaBa98] have been proposed in the literature to offer this control knob to the service providers. However, they lack the simplicity and ease of integration enjoyed by priority based schemes.

In this thesis, we propose and investigate an analytic approach to obtain a controllable service differentiation policy based on the concept of dynamic priority queueing. Our approach is free from the drawbacks of strict priority based schemes. It can offer the service provider the necessary control on resource allocation for meeting QoS guarantees and for meeting the desired service differentiation objectives. It also obviates the adoption of complex admission control and scheduling schemes and retains the advantages of simplicity and the ease of implementation of priority based schemes.

The proposed service differentiation policy is similar to certain threshold-based polling schemes as known in the queueing theory literature. One of the prime

contributions of this work is a unique solution to the resulting queueing model. All existing solutions [GoLu00], [HaId94], [BoKo95], [LeSe93] make an improper assumption of exponential service time distributions in the server. As shown in many web server related research efforts, this assumption is far from true in a practical web server. Therefore, we need to devise a new solution to the model. Boxman et. al. [BoDo97] derives a solution considering a general service time distribution in the server, but their solution is limited to only two classes of requests and involves Laplace-Stieltjes Transforms (LST). In contrast, we devise solutions for both two-class and multiclass scenario without assuming exponential service time distribution. We adopt the powerful matrix-geometric method to the solution, which does not involve LSTs and thus, useful numerical results required for system engineering can be obtained from the resulting analytical model.

We demonstrate the usefulness of our proposed approach in a practical scenario by conducting our analytic method on a real web trace. From the numerical results and analysis shown in the paper, we find that by adopting our method service providers can obtain a service differentiation policy with tunable control parameters, which they can set to appropriate values to ensure QoS guarantees and to reach desired service differentiation objectives.

This thesis is organized as follows: In Chapter 2, some relevant background and preliminaries related to the addressed problem and our proposed methodology are explored. Chapter 3 provides an overview of the related literature and distinguishes the contributions of the thesis. In Chapter 4, we formulate the problem considering two classes of traffic- QoS traffic and best-effort traffic and map the system into a two-queue model. Chapter 4 also provides the detailed solution approach for the resulting model. In

Chapter 5, we extend the system model to deal with multiple QoS classes of traffic along with a best-effort class. Chapter 5 also shows the steps for setting and optimizing control parameters of the service policy. Chapter 6 shows the results obtained from our analytical model when fed with real web traces and the validation of these analytical results by simulations. Chapter 7 states the conclusions, summarizes the contributions of this thesis and possible future research directions.

2 Background and Preliminaries

In this thesis, we make extensive use of some queueing theoretic techniques and concepts including Markov chains, matrix-geometric methods and phase-type service distributions. We also find that, QoS in web servers has some issues related to the choice of performance measures and traffic classification schemes. This chapter is aimed at exploring these background areas.

2.1 Markov Chains

Markov chains are an important type of stochastic process and are widely used to model many applications analyzed by queueing theory and many other branches of applied sciences and mathematics. This section aims to present a brief introduction, which would be useful for comprehending the queueing models in this thesis.

A stochastic process $\{X(t): t \geq 0\}$ with finite state space Ω is a family of discrete random variables taking values in Ω indexed by a continuous parameter t, thought of as time. Markov chain is based on a very important condition on this stochastic process, known as Markov property. Informally, the Markov property says that given the present, the past and the future are independent. Suppose that we know the value of the process at the k increasing time points $t_0 < t_1 < \cdots < t_{k-1}$ and we desire the conditional distribution of the process at time $t_k > t_{k-1}$. The Markov property says that this conditional distribution depends only on the most recently observed time and is independent of past events.

$$P\left(X(t_k)=i_k \mid X(t_0)=i_0, X(t_1)=i_1, \cdots, X(t_{k-1})=i_{k-1}\right)=P\left(X(t_k)=i_k \mid X(t_{k-1})=i_{k-1}\right)$$

However, it is important to note that Markov chain has historically been classified into two classes according to the considered nature of the time points: *discrete time Markov chains* (DTMC) and *continuous time Markov chains* (CTMC). We adopt a continuous time analysis in this thesis. Therefore, we shall focus on CTMC.

Each finite-state-space CTMC is completely described by an *initial probability vector* $p(0)$, whose ith element is the probability that the chain begins in state i at time 0, and an *infinitesimal rate (or generator) matrix* $Q=\{q_{ij}\}$. These two objects determine the distribution of $p(t)$, the vector of probabilities for each state at time t. If $d=|\Omega|$ then $p(t)$ will be a $1\times d$ row vector while Q is a $d\times d$ matrix.

The off-diagonal elements of Q are nonnegative and can be interpreted as the rate of transitions from i directly to j given the process is in state i. The diagonal elements of Q equal the negative row sums of the off diagonal elements, so that

$$q_{ii}=-\sum_{i\neq j} q_{ij}$$

This implies that the row sums of Q equal 0:

$$Q=\begin{pmatrix} -q_0 & q_{0,1} & \cdots & q_{0,j} & \cdots \\ q_{1,0} & -q_1 & \cdots & q_{1,j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \\ q_{i,0} & q_{i,1} & \cdots & q_{i,j} & \cdots \\ \vdots & \vdots & \cdots & \vdots & \cdots \end{pmatrix}.$$

The $d \times d$ *Markov probability transition matrix* $P(t) = \{p_{ij}(t)\}$ contains the probabilities that the chain is in state j at time t, given we began in state i for each pair of states. Transition matrices satisfy the expression

$$P(s + t) = P(s)P(t).$$

for each $s, t \geq 0$. We can also represent the transition matrices with a matrix exponential.

$$P(t) = e^{Qt} = \sum_{k=0}^{\infty} \frac{Q^k t^k}{k!}$$

for Q defined as above where we understand Q^0 to be the $d \times d$ identity matrix $\mathbf{I}$ with ones down the main diagonal and zeros elsewhere. We can write $p(t) = p(0)P(t)$.

The *dwell time* (waiting time) in state i before a transition to some other state is exponentially distributed with rate $q_i = -q_{ii}$. Given that a transition from i occurs, the probability that the transition is to state j is q_{ij} / q_i.

If we assume that the Markov chain is irreducible, meaning that it is possible to get from any state i to any state j in a finite amount of time, it follows that the Markov chain is positive recurrent, meaning that the expected return time for each state is finite. (For infinite state spaces, irreducibility need not imply positive recurrence.) These two conditions suffice to imply the existence of a unique stationary distribution $\mathbf{x} = \{x_i\}$. If $p(0) = xp(0)$, then $p(t) = x$ for all t. In addition, the limiting transition probabilities as time goes to infinity do not depend on the initial state and take the values of the steady state distribution.

$$\lim_{t \to \infty} P(t) = e.x.$$

Here e is a column vector consisting entirely of 1's and of the same dimension as of x. Notice that the last expression is a $d \times d$ matrix where each row is x. Finite-state-space irreducible continuous-time Markov chains are also called *ergodic*. We assert that ergodic Markov chains have a unique stationary distribution, and that this distribution may be found by finding the right-eigenvector of Q associated with eigenvalue 0, so that

$$x.Q = \mathbf{0}$$

where $\mathbf{0}$ is a row vector where all elements are 0.

An analysis of a system behavior described through Markov chains often involves finding stationary distributions for its states. Once these probabilities are obtained, different performance measures of interests can be derived.

We shall conclude this section by giving an example of a Markov chain that describes an M/M/1 queue. Requests arrive in the queue at a rate λ following a Poisson process, while the service time in the server is exponentially distributed with a service completion rate of μ. The state of the queue is described by the number of requests in the system. A new arrival increases the number in the system by 1 with a rate λ and a service completion reduces the number of customers in the system with a rate μ. Therefore, the rate matrix of the Markov chain will be as follows:

$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 & 0 & \cdots \\ \mu & -(\lambda+\mu) & \lambda & 0 & 0 & 0 & 0 & \cdots \\ 0 & \mu & -(\lambda+\mu) & \lambda & 0 & 0 & 0 & \cdots \\ 0 & 0 & \mu & -(\lambda+\mu) & \lambda & 0 & 0 & \cdots \\ 0 & 0 & 0 & \mu & -(\lambda+\mu) & \lambda & 0 & \cdots \\ \cdots & & & & & & & \cdots \end{pmatrix}.$$

Markov chains are also presented pictorially where a node represents a state and arcs show transitions among the states (Fig. 1).

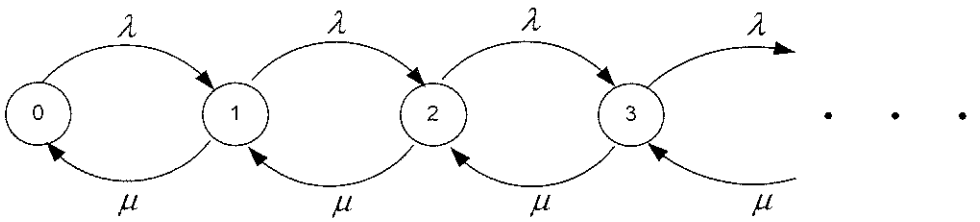

Fig. 1. Markov Chain for an M/M/1 queue

2.2 Phase-type Distribution

The queueing system discussed in the previous section has Markovian structure and is much more easily tractable analytically than systems involving non-Markovian parts. For the latter class of systems the mathematical machinery is more involved. A good property of the fully Markovian systems is that they can be mapped directly to a CTMC, and due to the memoryless property it suffices to use a single number to describe the state of the system and its future evolution. This single number is the number of customers or requests in the system. Unfortunately, it is a serious restriction to stick with Markovian systems. If the arrival or the service times are no more exponentially distributed, the Markov property does not hold anymore with a single variable for the state in the Markov chain. There are two principal ways to deal with other service time distributions:

- We may apply the formulas derived for the M/G/1 systems. However, if we want more information than mean values you have to work with Laplace transforms and their inverses. This can become quite cumbersome.
- Try to find a "somehow Markovian" approximation for the service time distribution in question.

The second way is indeed a valid one. The idea is to approximate an arbitrary service time distribution $B(t)$ by a so-called phase-type distribution, which can be generated by the operation of a Markov chain. For a detailed description on phase-type distribution see [Neut75], in which Marcel Neuts introduced its concept.

Given an arbitrary distribution function $B(t)$, the idea is to construct a finite CTMC with $n + 1$ states. A number $m > 0$ of these states are transient states, and the remaining $n+1-m > 0$ states are absorbing states. Furthermore, a subset of states is marked as start states. In general, a phase-type distribution is expressed by (β, S) where S is the rate matrix of transitions among the m transient states and β is the probability vector specifying the probabilities of starting the Markov chain at the corresponding start states.

The random variable generated by the model and used to approximate the given distribution is the time to reach one of the absorbing states from the chosen start state. The server of the given queueing system (with its arbitrary service time distribution $B(t)$) is replaced by a server with internal states. The transition process is just given by the CTMC, and at any time at most one customer is allowed to be in the CTMC. A customer starts at one of the start states (in case of multiple start states one of them is randomly selected) and proceeds through the transient states to one of the absorbing states. As soon

as an absorbing state is reached, the service is finished. The whole service facility is seen as a box, which can hold at most one customer and which has some internal state: we have to specify at which state of the CTMC the customer currently is.

To use this approach in queueing systems, we have to expand the state description of a queueing system: we have to specify the number of customers in the system as well as the current state of the box, i.e. the state of the server-internal CTMC where the customer in service currently is. The solution of systems where the state description consists of two or more components is often carried out with the matrix-geometric method

The main focus in the next section is on this solution method (the so-called matrix-geometric method) which allows solving queueing systems with phase-type distributions and certain other queueing systems for their steady state vector.

2.3 Matrix-Geometric Method

Matrix-geometric method, introduced by Marcel Neuts [Neut81], has been used as a powerful analytical tool for many queueing systems. The method is based on two ingredients: the first one is to use matrices and vectors instead of scalars, and the second one is a certain heuristic observation. We have seen this heuristic observation in the example presented in the previous section. The observation is that "geometrically looking" solutions are found quite often when the generator matrix Q shows some "regularity". The notion of a matrix comes into play, since the matrix-geometric method is developed for systems with vectorial state descriptions (i.e. where a state description has at least two elements.

As an example, consider the following fattened transition matrix, where we have used the abbreviation $a = 2\mu + \lambda$:

$$Q = \begin{pmatrix} -\lambda & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 2\mu & -a & 0 & \lambda & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 2\mu & -a & 0 & \lambda & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 2\mu & -a & 0 & \lambda & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 2\mu & -a & 0 & \lambda & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 2\mu & -a & 0 & \lambda & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 2\mu & -a & 0 & \lambda & \cdots \\ \cdots & & & & & & & & & \cdots \end{pmatrix}$$

This matrix obviously has a repetitive structure. Furthermore, by forming the sub-matrices:

$$B_{0,0} = (-\lambda);\ B_{0,1} = (-\lambda, 0)\ ;\ B_{1,0} = \begin{pmatrix} 2\mu \\ 0 \end{pmatrix}$$

$$B_{1,1} = \begin{pmatrix} -a & 0 \\ 2\mu & -a \end{pmatrix};\ B_{2,1} = \begin{pmatrix} 0 & 2\mu \\ 0 & 0 \end{pmatrix};$$

$$A_0 = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix};\ A_1 = \begin{pmatrix} -a & 0 \\ 2\mu & -a \end{pmatrix};\ A_2 = \begin{pmatrix} 0 & 2\mu \\ 0 & 0 \end{pmatrix}$$

we can write Q as a *matrix of matrices*:

$$\begin{pmatrix} B_{0,0} & B_{0,1} & & & & \\ B_{1,0} & A_1 & A_0 & & & \\ & B_{2,1} & A_1 & A_0 & & \\ & & A_2 & A_1 & A_0 & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}$$

(Please note that for the sake of clarity, we do not write the zero matrices appearing in the Q matrix, instead we keep the corresponding spaces blank).

In the general case, to which we focus our attention now, the matrix Q again is written as a matrix of matrices and has the following structure:

$$\begin{pmatrix} B_{0,0} & B_{0,1} & & & & & \\ B_{1,0} & B_{1,1} & A_0 & & & & \\ B_{2,0} & B_{2,1} & A_1 & A_0 & & & \\ B_{3,0} & B_{3,1} & A_2 & A_1 & A_0 & & \\ B_{4,0} & B_{4,1} & B_{4,2} & A_2 & A_1 & A_0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Roughly speaking, the matrices $B_{i,k}$ correspond to boundary states, and the matrices A_i correspond to repetitive states.

Now, we have to find a steady state vector $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots)$ (the $\mathbf{x}_i$ are vectors here), which solves:

$$x.Q = \mathbf{0}$$

$$x.e = 1$$

To find a general solution for the generator matrix, we then proceed by writing a balance equation for the repeating portion of the process:

$$\sum_{k=0}^{\infty} x_{j-1+k} A_k = \mathbf{0}, \quad j = 2, 3, \cdots$$

We then guess a matrix geometric solution

$$x_j = x_{j-1} R, \quad j = 2, 3, \cdots$$

or that

$$x_j = x_1 R^{j-1}, \quad j = 2, 3, \cdots$$

Subsequently, R solves

$$\sum_{k=0}^{\infty} R^k A_k = \mathbf{0}.$$

To determine a solution for the boundary states we must solve the following linear equations

$$(x_0, x_1)\begin{pmatrix} B_{0,0} & B_{0,1} \\ \sum_{k=1}^{\infty} R^{k-1} B_{k,0} & \sum_{k=1}^{\infty} R^{k-1} B_{k,1} \end{pmatrix} = \mathbf{0}\ .$$

These equations do not permit a unique solution and we must use the normalizing condition

$$\begin{aligned} 1 &= x.e \\ &= \sum_{i=0}^{\infty} x_i e \\ &= x_0 e + x_1 \sum_{i=1}^{\infty} R^{k-1} e \\ &= x_0 e + x_1 (I-R)^{-1} e \end{aligned}$$

After finding the x_0 and x_1, we can now determine the further steady state vector probabilities $x_2, x_3, \cdots$ by forming matrix powers of R

$$x_j = x_1 R^{j-1}, \quad j = 2,3,\cdots$$

2.3.1 Quasi-Birth-Death (QBD) Processes

A birth-and-death process allows only adjacent transitions. The example in section 2.1 is an example of a birth-and-death process. Adding the name "quasi" reflects that the nearest neighbor transitions are interpreted in terms of vector of states; state transitions

are only possible on the same level or between adjacent levels. So, Quasi-Birth-Death (QBD) processes will typically have the following form of Q matrix:

$$\begin{pmatrix} B & C & & & & & \\ E & A_{1,1} & A_{1,2} & & & & \\ & A_{2,1} & A_{2,2} & A_{2,3} & & & \\ & & A_{3,2} & A_{3,3} & A_{3,4} & & \\ & & & A_{4,3} & A_{4,4} & A_{4,5} & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}$$

However, there are two important classes of QBD processes: level dependent and level independent. In the level dependent scenario, the transition rate matrices among the neighboring state vectors change with the level of the vector states in the Markov chain. Level independent QBD processes, on the other hand, have repeated structure beyond the boundary states:

$$\begin{pmatrix} B & C & & & & \\ E & A_1 & A_0 & & & \\ & A_2 & A_1 & A_0 & & \\ & & A_2 & A_1 & A_0 & \\ & & & A_2 & A_1 & A_0 \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}.$$

Matrix-geometric method is directly applicable to such processes and the solution for the boundary states becomes:

$$(x_0, x_1) \begin{pmatrix} B & C \\ E & A_1 + R A_2 \end{pmatrix} = \mathbf{0}$$

and the solution to R matrix is more straight forward to obtain from the following equation:

$$A_0 + RA_1 + R^2 A_2 = \mathbf{0}.$$

Level dependent QBDs sometimes can be partitioned into level-independent QBDs and we can then apply the above solution procedure to each of the blocks. The results then can be combined to get the required steady state computations.

2.4 Quality of Service (QoS) in Web Servers

QoS is closely tied with the notion of perceived performances. However, there are two perspectives to view the scenario. The web service providers will obviously try to have maximum utilization of the limited web server resources they posses. On the other hand, the clients will be expecting the performance to be compliant with what they are paying for. So, in order to be competitive in the market, the providers also need to include in their objective the provisioning of the maximum satisfaction of the clients. Moreover, web servers have to handle different types of requests having different performance constraints that have to be met to keep the service meaningful. As a result, QoS provisioning in web servers entails two distinct issues: performance criteria and classification policies.

Throughput and utilization are two important benchmarks for measuring the performances of web servers. Throughput is defined as the number of requests

successfully served per unit time by the server. Utilization refers to the fraction of the time the web server is busy serving requests. However, the response time is likely to be the main performance criteria the clients will be interested in. Response time is the time period between the moment a request is submitted by a client and the moment when the request is fully served by the server. The trend in the web server technological development suggests that, when several classes of traffic are handled by the web servers, the focus of web server performance is shifting from throughput and utilization benchmarks [NaBa99] [BaCr99] [NiGe97] to guaranteeing delay bounds i.e. the response times for these different classes of requests [AbSt01] [KaKn00] [EgHe99] [AlMa98] [ChMo99] [BhFr99]. These guarantees of response times are often expressed in terms of statistical averages.

The issue of traffic classification in web server QoS schemes may be accomplished by different classification policies. The policies differ widely depending on the objectives of the service provider as well as the types of contents provided. It is often a common practice to host different web sites in a single web server facility, because it is more cost effective both for the server owner and the web site owners. Such shared hosting facilities then resort to the classification of the request objects hosted in them depending on the object ownerships. In general, the classification policies, however, encompass a much wider scope. The following list provides some examples of classification policies that can be implemented:

- Source-based: Some web servers may try to differentiate among the incoming requests depending on the origin of the requests. The policy might be to facilitate better service for some organization or geographical area. The

classifier looks into the incoming HTTP or other application level packets to determine this information and puts the requests into the appropriate classes.

- Content-based: Objects are usually stored in the server as files. The file types can vary from normal HTML to different multimedia types such as pictures. Servers have to handle dynamic objects as well. These objects are created in the server by some programming interface or scripts triggered by some user requests. Serving some objects might have to be done with higher priority than some others. For example, credit card transactions might be given more importance than the browsing request in an online shopping store. The classifier in this case has to analyze the urls requested and find out the type of files being requested.
- Economics-based: In shared hosting centers, payment by object owners in the server might determine the classification of the requests. In some cases, the hosting center might reach some agreement on the service levels with the site owners. The agreements will detail the payment structure on the basis of the quality or level of service expected. The classifier will have the information about the ownership of the objects hosted in the server and will do the classification on the basis of the ownerships.
- Popularity-based: Object popularity often drives the classification policies in many web servers. To have greater client satisfaction, the more popular objects should be given higher service priority than the less popular ones. Popularity is often a dynamically changing attribute for some particular

object. However, the classifier can maintain separate pools of requests on the basis of the ranges of popularities on the entire collection of objects.

- Size-based: The sizes of the requested objects might be used for request classification in some web server scenario. Assigning priorities to objects on the basis of their sizes has impact on the overall performance of the server. If bigger objects are given higher priority, then requests for the short sized objects might suffer unwanted delays. On the other hand, serving larger sized file with more priority may enhance the server throughput.

However, it is important to note that, the term QoS can have much wider scope of meaning other than the above discussed performance parameters. Diverse issues such as security, authenticity can also be brought into the scope of negotiated QoS. This thesis focuses on the performance aspects of QoS and will thus formulate the objectives accordingly. Another important point we should make clear here is that response times should also include any queueing and transfer delay along the communication link or path. However, since the overhead incurred through the communication channel cannot be controlled by the web server, we do not consider it for inclusion in the response times of the web server.

3 Related Work

The emergence of World Wide Web has fueled immense enthusiasm in the research community. The challenges remain in two main fronts. First, the web servers are the central entities for delivering the contents and their resources have to be appropriately utilized. The other front is the Internet which ultimately brings the content to the user premises. Considerable research has been ongoing in both of these fronts to incorporate QoS aspects or to enable service differentiation among different web traffics. This research, however, concentrates on the techniques for server resource utilization. The approach taken in this thesis heavily depends on an analysis and solution to a queueing theoretic problem which has also drawn considerable attention from the queueing theorist community. The following sections will highlight the research efforts in both of these areas.

3.1 Service Differentiation and QoS Aware Resource Allocation in Web Servers

Providing differentiated services in web servers has been investigated in a number of research efforts. In [AlMa98], the authors investigate simple priority based scheduling schemes in the web servers to see the performance benefits achieved by the premium service clients. They propose a notion of quality of service by associating priorities with requests from different sites. The HTTP server schedules requests according to priorities,

thereby ensuring that preferred sites (with higher priority) are allocated resources before other sites. They experiment by incorporating modification into the server process or the operating system according to the priority schemes to handle different classes of requests differently.

Bhatti and Friedrich [BhaFri99] modified the Apache Web server to support tiered Web services. A connection manager intercepts incoming HTTP requests and classifies them into different QoS classes. The classified requests are placed into appropriate tier (QoS class) queue. The admission policy decides if a request should be rejected or not. The scheduling policy decides the way how a work process services an admitted request. What kind of policies to use is configurable through a policy management interface, including a network fabric API.

Eggert and Heidemann [EgHe99] proposed a set of application-level-only control techniques to support two levels of Web service priorities: a foreground request class and a background request class, where a background request is defined as a low priority pre-emptable transaction whose presence in the system never decreases the performance of concurrent foreground request. Three application-level mechanisms were used to slow down the background request class: limiting the process pool size, decreasing the process scheduling priority value for the processes dedicated to the background request class, and finally throttling the network transmission rate of the processes for the background request class.

Pandey et al. [PaBa98] presented a QoS model for HTTP servers which aimed at enabling a site to customize how the server should response to external requests. Pages or set of pages are associated with QoS constrains, and a request can be served only when

no constrain is not violated. A dedicated QoS daemon sits behind the Web server nodes to determine that a node should service, reject or forward the request it just received according to the load on each node and each service calss's current resource (network bandwidth) consumption. In this system, requests are not scheduled or re-ordered. They are either admitted or rejected.

However, just providing different service levels to different classes are often not enough. Delivering the promised quality of service for the appropriate class of requests is often the objective and some controlling parameters have to be in hand to regulate the server sharing to enable this. None of the above works address this aspect of service differentiation. Their differentiation models in these works do not provide a "tuning knob" to control the performance distance between different classes.

Control theoretic approach has been adopted by some researchers to ensure guaranteed service in web servers for certain class of clients [AbSh02] [AbSt01] [AbSh99]. The core of the approach is the introduction of feedback control architecture for adapting resource allocation in the web server. They formulate the adaptive resource allocation problem as one of feedback control and apply feedback control theory to develop the resource allocation algorithm. To enforce relative delays among service classes they propose to use a feedback control loop to reallocate the number of server processes for each service class. They implement and evaluate the adaptive architecture on a modified Apache web server.

Chen and Mohapatra [ChMo02] propose a technique that allocates requests to Apache server processes to minimize per-class response time bounds. A weight is assigned to each class of requests to maximize a "server productivity" function, defined in terms of

the number of pending requests in each class, a (fixed) per-class processing-time requirement, and a (fixed) delay bound. The paper is very unclear on implementation details and how the assigned weights are actually employed. This paper considers only static Web pages and silently drops requests if delay bounds are exceeded.

Kanodia and Knightly [KaKn02][KanKni00] develop an approach based on the use of service envelopes that capture the server's request rate and service capacity without requiring detailed modeling techniques. The admission control technique attempts to meet response-time bounds for multiple classes of service requests, so is more closely tied to the kind of SLAs (Service Level Agreements) that real systems may employ. However, the technique is only studied under a simple simulation of Web server behavior.

Lee et. al. [LeLu04] propose admission control algorithm and dynamic adaptation mechanism of class assignments to the requests in order to maintain a predefined ratio of performance among classes of web traffic. They intend to maximize the overall profit earned from the services where the pricing is proportional to this predefined ratio. They formulate the problem as a similar one to the well known kanpsack problem and the core to their algorithm is a heuristic similar to the ones used to solve this traditional problem. Similar algorithms were already proposed in [DoRa99] for packet scheduling in communication networks. They also propose a distributed algorithm based on game theoretic approach, where uncooperative clients can adapt the classes of their requests.

Unlike these approaches, we adopt a queueing theoretic approach to find the optimal allocation of the web server to classes of requests while guaranteeing the pledged QoS. We provide a powerful analytical tool to assist the server assignment process. Most of the

related research mainly focused on architectural aspects of QoS aware web server design and related implementation issues and used simulation or prototype implementation to analyze the performance behavior. Theoretical analysis for the proposed methods remained unattended. In contrast, we focus on a controllable service differentiation policy, and develop analytical method to find the desired values of the key parameters for it. Using our method, the server performance behavior can be analyzed depending on these parameters in a view to reach the performance objectives.

3.2 Threshold-based Polling and Priority Queueing

The developed service differantiation policy maps into a queueing problem largely related to the threshold-based polling and priority queueing schemes, as known in the queueing theory literature. A considerable amount of research has been done in this area as well. In next few paragraphs, we shall focus on some notable related works to this queueing theoretical aspect of our work.

Some threshold-based polling schemes are proposed and analyzed by Lee and Sengupta [Lee93][LeSe93], Havekort et. al. [HaId94], Boxma et. al [BoKo95][BoKo95a][DoBo95]. In [Lee93], a single server two queue model is considered where the high priority queue is served exhaustively and the low priority queue receives k-limited service. In [LeSe93], the server serves both of the queues alternatively unless the high priority queue has exceeded a certain threshold and after that the server continues to serve the high priority queue until its queue length is back to the threshold point. In [HaId94], the same model is analyzed using stochastic Petri nets and a

variant is also suggested and analyzed where once the threshold in the high priority queue is reached the server serves exhaustively until the high priority queue becomes empty. Boxma et. al analyzes the same model considering preemptive service switching in [BoKo95] and non-preemptive service switching in [BoKo95a]. Techniques for computation of bounds for performance measures of multi-server threshold-based queueing systems with hysteresis and non-instantaneous server activation are given in [GoLu00]. All of these works consider exponentially distributed service time in the server, which is vastly an impractical assumption for processing times in a web server. Instead, we solve the two-queue model and then extend it for multiclass scenario considering phase-type service time distributions, which can virtually abstract all the practical service time distributions in the server. Our process is much more elegant because we utilize the nice algorithmic approach of Matrix-geometric methods instead of complex transform based analysis. Being free from LST (Laplace-Stieltjes Transforms), our results allow us to numerically compute the distributions of queue length and other performance measures with greater ease.

The other related area, which is priority queueing in general, is a very old topic in queueing theory and there are dozens if not hundreds of published articles on this analysis. Interested readers can refer to [Mill60] for a detailed analysis of priority queueing techniques.. However, multiple priority queue model has not been studied well compared to its two queue counter part and there have been mostly transformation based solutions approaches to priority queue models. Transform based methods are mathematically well furnished, but disadvantageous for numerical result. With respect to performance measures, often make it extremely difficult to get beyond the mean values of

queue lengths. Alfa et. al. [AlLi03] and Gai et. al. [GaHa88], developed matrix-geometric solution method for multiclass MAP/Ph/1 priority queues. However, the queues in consideration have strict priorities without any notion of thresholds to control the level of services they receive. In this thesis, our analysis includes multiple priority classes with threshold controlled service policy. We allow phase-type service time distributions in all of the queues. To the best of our knowledge, our work is the first solution to multiclass priority queueing system with threshold-based polling service policy that allows phase-type service time distribution in the server.

4 Two-Queue System Model

In this chapter, we consider a single web server which handles two classes of traffic, namely QoS requests and best-effort requests. The QoS requests are those for which a statistical guarantee of overall performance is pledged from the service provider. The rest of the traffic treated in the other queue is best-effort, which means the service provider does not guarantee any performance but tries to offer the best for increased client satisfaction (which is important from business point of view), provided the QoS guarantee in the other class is met. The scenario of the system is depicted in Fig. 2. The admission controllers in both of the queues only admit requests if they do not violate the arrival rates decided apriori. The decision can be based on the service agreements or the

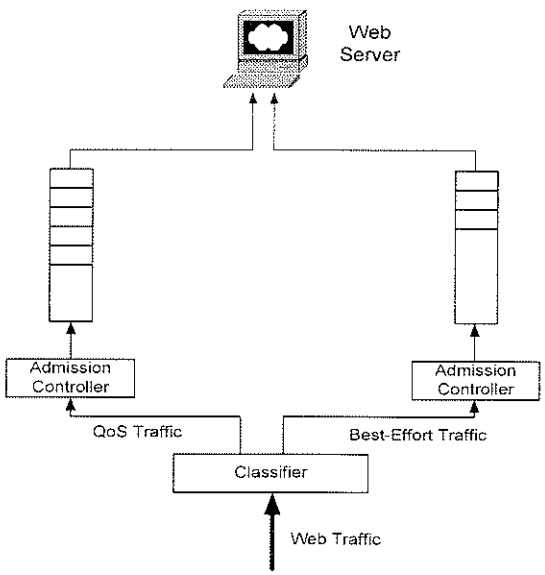

Fig. 2. QoS-aware web server model serving QoS and best-effort traffic

decision made by the server owner. These same rates are also input to our analytical model which seeks to find appropriate parameter value of the service differentiation, assuming that these rates are pre-decided. The use of our analytical model makes the admission controller implementation a very simple one, since it can just admit requests on the basis of the known allowable arrival rates.

4.1 Differentiated Service Policy

To provide the appropriate treatment of the two classes of traffic the server follows service policy as follows. At the time the service is ongoing in any of the queues, requests may be queued up in both of the queues due to arrivals of new requests. The server adopts variable and different amount of time treating these different queues i.e. the server is time-shared. The server tries to treat the QoS traffic with higher priority, but does not want the best-effort traffic to be starved indefinitely. On the other hand, it also does not want to serve the best-effort traffic so long as to risk the QoS traffic being deprived of pledged performance. Therefore, it puts a maximum limit on the number of waiting traffic of the QoS class while serving the best-effort queue. When this limit is reached, the server stops serving the best effort traffic and starts to serve the QoS queue. However, the switching is non-preemptive i.e. the server finishes the current request in service before it goes to the other queue. From the QoS queue perspective, once the server is treating the QoS traffic, the nature of the service will be exhaustive i.e. the server will switch to the best-effort traffic only after there is no outstanding request in this queue.

What follows in the following section is how we map the operation of this server with this serving policy into a queueing model that will be analyzed then.

4.2 Mapping the System to a Queueing Model

Given the above description of the server model and the service policy we now proceed to map a queueing model to it. The resulting queueing model is that of threshold-based polling systems seen in many queueing system applications. In such a model with two queues, a single server serves the queues following a threshold-based policy. Queue 1 enjoys higher priority than queue 2. We map the QoS queue to queue 1 and the best effort queue as queue 2 of this model. From now, by class 1 and class 2 requests we shall refer to QoS requests and best-effort requests respectively. When the system is empty, it waits in queue 1. When a new request comes in when the system is empty, the server starts serving in the corresponding queue. The service in the queue 1 is exhaustive. However, the service in queue 2 is interrupted if the requests queued in queue 1 reaches a predefined threshold T. In this case, the server switches from queue 2 to queue 1. The switching policy is non-preemptive i.e. the server switches to queue 1 only after finishing the request currently in service. In the analysis we assume queue 1 has infinite buffer, while the queue 2 has finite buffer capacity of L. Due to this finite capacity of queue 2, some best-effort request will be dropped without getting service. Loss probability in the

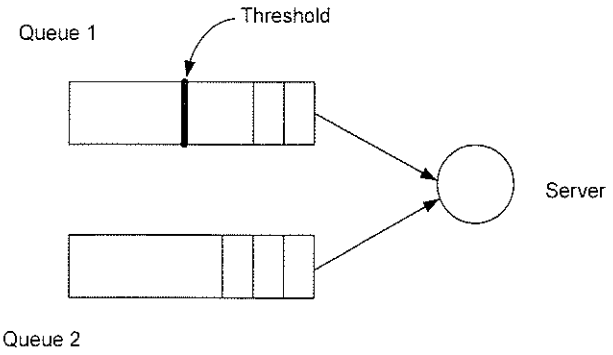

Fig. 3. Queueing model of threshold-based polling in two queues

queue 2 can be treated as a performance measure for the best-effort requests, because for keeping the guaranteed service of the QoS queue some best-effort requests might have to be dropped. We might also seek to minimize this loss probability in our objective.

4.3 Objective Formulation

Given the above service policy, the threshold set at the QoS queue constitutes the control parameter. In other words the service provider has a control parameter, which he needs to set to appropriate value. To accomplish this, we have to express the performance measures explicitly in terms of the threshold T. This threshold can only be set to those values, where the resulting performance measures conform to the performance guarantees. For meeting the optimization objective, these guarantees are treated as constraints and the threshold is tuned to optimize the objective function while meeting these constraints.

4.4 Arrival and Service Process

The arrival in both of the queues occurs following the Poisson arrival process: the rate of arrival in queue 1 is λ_1 and that in the queue 2 is λ_2. The rationale behind choosing the Poisson arrival process is the evidence cited in web traffic related literature that requests incoming to the web servers often follow exponential inter-arrival times. However, for service time distribution we use continuous phase-type distribution for both classes of requests. Though the service times are often assumed to be exponentially distributed as well, the practical scenario in web servers deviates largely making the resulting analysis impractical. To make our analysis applicable to real systems, we resort to use phase-type distribution introduced by Neuts [Neut75]. The beauty of the phase-type distribution is that it can model almost all the probabilistic distribution we encounter in our practical life. In our analysis, service time in both of the queues follow the same phase-type distribution expressed by (β, S), with dimension n. Here, n is the number of phases in the service before completion, β is the initial probability vector with n elements representing the probability of service initiation in the corresponding phases and S is the transition rate matrix among the phases.

4.5 Markov Chain Analysis

For the described model of the system, a full state description requires the knowledge of the number of requests in both queue 1 and queue 2, which class of request the server

is currently serving and the phase of the service. We follow a layered approach for describing the system states as adopted by Alfa [Alfa98]. The top layer describes the transitions among the number of requests in queue 1 (this number is also the *level* of the process), the middle layer involves transitions among the number of requests in queue 2 and the bottom layer represents the transition among the phases of the service. The complete state space can be described formally as follows:

- $\Delta_0 = \{(0)\}$
- $\Delta_1 = \{(0, j, m) | j = 1, \cdots, L; m = 1, \cdots, n\}$
- $\Delta_2 = \{(i, 0, m) | i \geq 1;\ m = 1, \cdots, n\}$
- $\Delta_3 = \{(i, j, u, m) | i \geq 1;\ j = 1, 2, 3, \cdots, L; u = 1, 2; m = 1, 2, \cdots, n\}$

State space $\Delta = \Delta_0 \cup \Delta_1 \cup \Delta_2 \cup \Delta_3$

The state Δ_0 refers to an empty system. In states $(0, j, m) \in \Delta_1$, there are only class 2 requests in the system $(1 \leq j \leq L)$ with the service in the m-th phase for the request being serviced currently. States represented by $(i, 0, m) \in \Delta_2$ refer to those states where there are only class 1 requests in the system $(i \geq 1)$ with the service in the m-th phase. Finally, in states $(i, j, u, m) \in \Delta_3$, there are at least one class 1 request in the system $(i \geq 1)$ and j class 2 requests also in the system, the server is currently serving class u request $(u = 1, 2)$ with the phase of the service currently being m.

We now proceed to describing the generator matrix for the associated CTMC. The generator matrix Q has the following form:

$$
Q = \begin{array}{c} 0 \\ 1 \\ 2 \\ \vdots \\ T-1 \\ T \\ \vdots \end{array}
\begin{pmatrix}
B & C & & & & & & \\
E & A_1 & A_0 & & & & & \\
 & A_2 & A_1 & A_0 & & & & \\
 & & \ddots & \ddots & \ddots & & & \\
 & & & A_2 & A_1 & A_0 & & \\
 & & & & A_2 & A_1^{*} & A_0 & \\
 & & & & & \ddots & \ddots & \ddots
\end{pmatrix}
$$

The matrix B, C and E are non-repeating and will be described first. At level 0, all that needs to be specified is the number of class 2 requests and the service phase of the class 2 request in service. Level 0 must also accommodate an idle state, which occupies the top row of Q. A class 2 request arriving to an idle system enters directly into service, whereas all other such arrivals increase the number of class 2 requests by one. As long as there are class 2 requests and no class 1 request, the server keeps selecting from queue 2. Changes in the phase of service are the only transitions not resulting in a change in the number of class 2 requests.

Consequently, the matrix B, which has dimension $(1+nL)\times(1+nL)$, is defined as:

$$
B = \begin{pmatrix}
-\lambda & \lambda_2\beta & & & \\
S^0 & S-\lambda I(n) & \lambda_2 I(n) & & \\
 & S^0\beta & S-\lambda I(n) & \lambda_2 I(n) & \\
 & \ddots & \ddots & \ddots & \\
 & & & S^0\beta & S-\lambda_1 I(n)
\end{pmatrix}.
$$

where $I(n)$ denotes the identity matrix of dimension n.

Matrix C describes arrival of class 1 request when the system is at level 0. So, C with dimension $(1+nL)\times(n+2nL)$ can be written as:

$$C=\begin{pmatrix} \lambda_1\beta & & & & \\ & M_1 & & & \\ & & M_1 & & \\ & & & \ddots & \\ & & & & M_1 \end{pmatrix}$$

where M_1 is a $n\times 2n$ matrix, defined as

$$M_1=\begin{pmatrix}\mathbf{0} & \lambda_1 I(n)\end{pmatrix}$$

Considering transitions from level 1 to level 0, we can now define the block matrix E. The necessary circumstance for these transitions is that with the completion of service for the class 1 request, the server shifts from queue 1 to queue 2, if there is any class 2 request. Therefore, the block matrix E of dimension $(n+2nL)\times(1+nL)$ is defined as:

$$E=\begin{pmatrix} S^0 & & & & \\ & M_2 & & & \\ & & M_2 & & \\ & & & \ddots & \\ & & & & M_2 \end{pmatrix}.$$

where M_2 is of dimension $[2n\times n]$ and defined as

$$M_2=\begin{pmatrix} S^0\beta \\ \mathbf{0} \end{pmatrix}.$$

Now, we turn our attention to describing the matrix A_1 that repeats up to level T-1 and the matrix A_1^* that starts repeating from level T. In fact, the former one converts into the later one from the point the threshold in the queue 1 has been reached. Both of them describe all transitions in which the level does not change. This includes the arrival of a class 2 request, the service completion of a class 2 request, and changes in the phase of service without a departure. The level remaining same means no arrival occurs for class 1 request. Note that when we consider matrix A_1 which repeats until level T-1, the threshold level in queue 1 has not been reached. As a result, in case of service completion of a class 2 request, contrary to a normal non-preemptive priority queueing system, the server will not switch to the class 1 queue if there is any class 2 request still waiting. However, if queue 2 becomes empty, the longest waiting class 1 request (if there is any) will enter the service.

Consequently, the remaining layers of A_1 are given by

$$A_1 = \begin{pmatrix} S-\lambda I(n) & M_3 & & & & \\ M_4 & M_5 & \lambda_2 I(2n) & & & \\ & M_6 & M_5 & \lambda_2 I(2n) & & \\ & \ddots & \ddots & \ddots & & \\ & & & & M_6 & M_5^* \end{pmatrix}.$$

where M_3, M_4, M_5, M_5^* and M_6 have dimensions respectively $n\times 2n$, $2n\times n$, $2n\times 2n$, $2n\times 2n$, $2n\times 2n$ and are defined as following:

$$M_3 = \begin{pmatrix} \lambda_2 I(n) & \mathbf{0} \end{pmatrix}$$

$$M_4 = \begin{pmatrix} \mathbf{0} \\ S^0\beta \end{pmatrix}$$

$$M_5 = \begin{pmatrix} S - \lambda I(n) & \mathbf{0} \\ \mathbf{0} & S - \lambda I(n) \end{pmatrix}$$

$$M_5^* = \begin{pmatrix} S - \lambda_1 I(n) & \mathbf{0} \\ \mathbf{0} & S - \lambda_1 I(n) \end{pmatrix}$$

$$M_6 = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & S^0\beta \end{pmatrix}.$$

In case of matrix A_1^*, the threshold level in queue 1 has already been reached. As a result, in case of a service completion of a class 2 request, the server will definitely switch to queue 1 from queue 2 even if there is any class 2 request waiting. Following description of A_1^* reflects this behavior; it differs from A_1 by the replacement of M_6 by M_7.

$$A_1^* = \begin{pmatrix} S - \lambda I(n) & M_3 & & & & \\ M_4 & M_5 & \lambda_2 I(2n) & & & \\ & M_7 & M_5 & \lambda_2 I(2n) & & \\ & \ddots & \ddots & \ddots & & \\ & & & & M_7 & M_5^* \end{pmatrix}$$

where M_7 with dimension $2n \times 2n$ is defined as:

$$M_7 = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ S^0\beta & \mathbf{0} \end{pmatrix}.$$

Block matrix A_2 corresponds to the service completion of a class 1 request followed immediately by the start of service of another. Necessarily from the nature of CTMC, there can be no change in the number of class 2 requests in queue 2. Note that A_2 continues to repeat even at level T and afterwards. This can be explained easily if we consider the exhaustive nature of the service in queue 1. Once the server is processing request of class 1, it will not switch to queue 2 until queue 1 becomes empty. It remains true regardless of whether the current level is higher or lower than the threshold value T.

Consequently, matrix A_2 with dimension $(n+2nL)\times(n+2nL)$ is defined by

$$A_2 = \begin{pmatrix} S^0\beta & & & & \\ & M_8 & & & \\ & & M_8 & & \\ & & & \ddots & \\ & & & & M_8 \end{pmatrix}.$$

where M_8 with dimension $2n\times 2n$ is defined by:

$$M_8 = \begin{pmatrix} S^0\beta & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

The remaining matrix is A_0 and corresponds to the arrival of a class 1 request. From the property of CTMC, no service completion or phase change can occur at the same

time. This also explains that A_0 will continue to repeat at level T and afterwards regardless of whether the threshold level is reached or not.

Consequently, matrix A_0 with dimension $(n+2nL)\times(n+2nL)$ is defined by

$$A_0 = \begin{pmatrix} \lambda_1 I(n) & & & & \\ & \lambda_1 I(2n) & & & \\ & & \lambda_1 I(2n) & & \\ & & & \ddots & \\ & & & & \lambda_1 I(2n) \end{pmatrix}.$$

4.6 Matrix-Geometric Solution for Steady State Distributions

We can see from its structure that the matrix Q is of a QBD process and matrix-geometric method [Neut81] can be applied for the analysis. However, the problem here is that the QBD is not a fully level-independent one. In fact, there is a potentially very big boundary block matrix in its upper left corner comprising levels starting from 0 to T-1. Consequently, the results from standard matrix-geometric methods for a homogenous infinite state QBD cannot be applied directly. However, the big boundary block has a nice repeating structure in itself and is followed by an infinite level repeating structure typical to a homogenous QBD; both of these will facilitate the analysis to a great extent. What follows now is the solution procedure that can be applied for this special structured Q matrix.

We can think of the Q matrix as consisting of two blocks: the finite dimension Block_1 covering up to level T-1 and the infinite dimension Block_2 covering level T and afterwards. In order to calculate the equilibrium distribution we first have to evaluate two matrices: the matrix $\tilde{R}$ for the Block_1 and the matrix R for the Block_2.

The subsequent analysis proceeds with the assumption that the system is stable i.e. Markov chain represented by Q is positive recurrent. It is intuitive that the infinite block Block_2 has the ultimate potential to drive the system into instability. Consequently, for stability it is sufficient to hold that

$$\pi A_0 < \pi A_2 \mathbf{1}, \text{ where } \pi = \pi A^*, \pi \mathbf{1} = 1, \text{ and } A^* = A_0 + A_1{}^* + A_2 \ .$$

We start by writing the finite block Block_1 as a stochastic matrix as follows:

$$\text{Block}_1 = \begin{pmatrix} B & C & & & & & \\ E & A_1 & A_0 & & & & \\ & A_2 & A_1 & A_0 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & & A_2 & A_1 + RA_2 \end{pmatrix}$$

The matrix $\tilde{R}$ can be found as the minimum non-negative solution to

$$\mathbf{0} = \mathrm{A}_0 + \tilde{R}A_1 + \tilde{R}^2 A_2$$

And the matrix R can be found as the minimum non-negative solution to the equation:

$$\mathbf{0} = \mathrm{A}_0 + RA_1{}^* + R^2 A_2$$

In the subsequent analysis, we will denote by $e(v)$ a column vector of 1's with dimension v. Let the steady state vector covering Block_1 be $[x_0', x_1', \cdots, x_{T-1}']$. As we wrote Block_1 as stochastic, we proceed by writing the following:

$$x_0' B + x_1' E = \mathbf{0}$$

$$x_0' C + x_1' \left(A_1 + \tilde{R} A_2 \right) = \mathbf{0}$$

$$x_0' e(1 + nL) + \sum_{i=1}^{T-1} x_i' e(n + 2nL) = 1$$

After solving the above systems of equations, we find the vectors x_0' and x_1'. Then we can express the rest of the steady state vector as

$$x_i' = x_1' \tilde{R}^{i-1}, \quad \text{for } 2 \le i \le T-1$$

Let, the steady state vector covering Block_2 be $[x_T', x_{T+1}', \cdots]$, considering Block_1 as stochastic. Then we can write

$$x_j' = x_{T-1}' R^{j-T+1}, \quad \text{for } j \ge T$$

Now, in order to find the actual steady state vector $x = [x_0, x_1, x_2, \cdots, x_{T-1}, x_T, x_{T+1}, \cdots]$, we normalize the computed steady state vectors by the following

$$\xi = x_0' e(1 + nL) + \sum_{i=1}^{\infty} x_i' e(n + 2nL)$$

After some simple algebraic manipulations, ξ can be written as:

$$\xi = x_0' e(1 + nL) + x_1' \left(\sum_{i=1}^{T-1} \tilde{R}^{i-1} + \tilde{R}^{T-2} R (I - R)^{-1} \right) e(n + 2nL)$$

The steady state vector x is found as follows:

$$x_i = \frac{x_i'}{\xi}, \quad \text{for } i \ge 0$$

4.7 Computing Performance Measures

After we derive the steady state distributions of the Markov chain, we now resort to computing the performance measures in this subsection.

4.7.1 Computing Queue Lengths and Loss Probability

Let $q_i(v)$ be the probability that there are i class v requests in the system, $v=1,2$. For the class 1 requests, we have

$$q_i(1)=\boldsymbol{x}_i e(n+2nL), \quad \text{for } i\geq 1$$

$$q_0(1)=\boldsymbol{x}_0 e(1+nL)$$

As the space for class 2 requests is limited to L, there could be at most L class 2 requests in the system and we can write

$$q_i(2)=\boldsymbol{x}_{0,i}\, e(n)+\boldsymbol{x}_1\left(\sum_{h=1}^{T-1}\tilde{R}^{h-1}+\tilde{R}^{T-2}R(I-R)^{-1}\right)\gamma_i, \text{ for } 1\leq i\leq L$$

$$q_0(2)=\boldsymbol{x}_{0,0}+\boldsymbol{x}_1\left(\sum_{h=1}^{T-1}\tilde{R}^{h-1}+\tilde{R}^{T-2}R(I-R)^{-1}\right)\gamma_0'$$

where $\gamma_i=\sum_{v=n(2i-1)+1}^{n(1+2i)} e_v$ and $\gamma_0'=\sum_{v=1}^{n} e_v$. The column vector e_v is of dimension $(n+2nL)$ with one at the v th position.

One of the performance measures, *the loss probability for class 2 requests* = $q_L(2)$

Let L_v be the mean number of class v requests in the system, $v=1,2$, then

$$L_1 = \sum_{i=1}^{\infty} i q_i(1)$$

$$= x_1 \left(\sum_{i=1}^{T-1} i\tilde{R}^{(i-1)} + \tilde{R}^{(T-2)} \sum_{i=T}^{\infty} i\, R^{(i-T+1)} \right) e(n+2nL)$$

$$= x_1 \left(\sum_{i=1}^{T-1} i\tilde{R}^{(i-1)} + \tilde{R}^{(T-2)} \left(TR(I-R)^{-1} + R^2 (I-R)^{-2} \right) \right) e(n+2nL)$$

$$L_2 = \sum_{i=1}^{L} i\, q_i(2)$$

4.7.2 Computing the Mean Response Times

Response time includes the time a request spends waiting in the queue plus its service time. After the mean queue lengths are known, finding mean waiting times is now possible by applying the Little's Law. Let $\overline{\tau}_v$ denote the mean response time for the class v requests. Then

$$\overline{\tau}_1 = \frac{1}{\lambda_1} L_1$$

Special care has to be taken for calculating mean waiting times for class 2 requests because the space for them is limited in the queue. Any class 2 request finding the queue 2 full will be lost. So, we have to consider the effective arrival rate $\lambda_2^* = \lambda_2 (1 - q_L(2))$. Then the mean response time for class 2 requests is

$$\bar{\tau}_2 = \frac{1}{\lambda_2^*} L_2$$

4.8 Setting the Threshold

Once the relationships between the performance measures and the threshold are found out using the above procedure, finding the appropriate values for this threshold is also possible. The relationships can be fitted to appropriate functions by using regression analysis. Now, the service provider may want to reach an optimization objective in addition to meeting the performance guarantee for QoS requests and have the optimization functions defined in terms of the performance measures. Because, the performance measures are explicitly expressed in terms of the threshold by our analytic method, the objective functions can now be defined in terms of this parameter as well and some non-linear constrained optimization techniques can be applied to find out its optimal value.

Let us consider the optimization objective to be the one in which we want to minimize the mean response time of the best-effort requests, while maintaining the guaranteed mean response time for the QoS requests. We can write the optimization problem formally as follows.

$$\begin{aligned} \text{minimize} \quad & \bar{\tau}_2(T) \\ \textit{subject to} \quad & \bar{\tau}_1(T) \quad \leq \delta_1 \\ & T \geq 0 \end{aligned}$$

where $\bar{\tau}_i(T)$, stands for the mean response time of the requests served in the ith queue $(i = 1,2.)$, δ_1 is the corresponding guaranteed mean response time for QoS requests and

the T inside the brackets indicates that the mean response times are functions of this threshold. The functions are likely to be non-linear ones and can be fitted by using regression from sample data collected from numerical results from our analysis. Then, non-linear constrained optimization techniques can be applied to find the sought optimum points [Murty95].

5 Model Extension for Multiple QoS Classes

In this chapter, we extend the model to consider multiple QoS classes of incoming requests. For a general multiclass model, we deal with K queues where $K-1$ of them are holding respective QoS class requests and the rest is holding the best-effort requests (Fig. 4). Therefore, we consider the Kth queue as the best-effort queue. Queue 1 is considered highest priority queue as before with priority decreasing along the increasing queue numbers. Chapter 4 provided a detailed analysis for $K = 2$. This chapter provides a general solution approach for cases where $K \geq 2$. Apart from chapter 4, we also consider the more general case, where the phase-type service time distributions of the different QoS classes have different parameters and are represented by (β_i, S_i), $1 \leq i \leq K$. However, to avoid unnecessary computational complexity, we keep their dimensions same which is m. Define that $S_i^0 = e - S_i e$, where e is a column vector of 1's of dimension m. We keep the arrival process for each class of request as Poisson with rates λ_i, $1 \leq i \leq K$. We also consider each of the queues to have infinite capacity. The service remains non-preemptive.

5.1 Service Policy for Multiclass Scenario

In this section, we extend the service policy described in chapter 4 to handle multiple classes of QoS requests. Each of the queues for QoS requests will have its own threshold. Let T_i represent the threshold for class i requests. So, there will be K-1 threshold points, $(T_1, \cdots, T_{K-1})$.

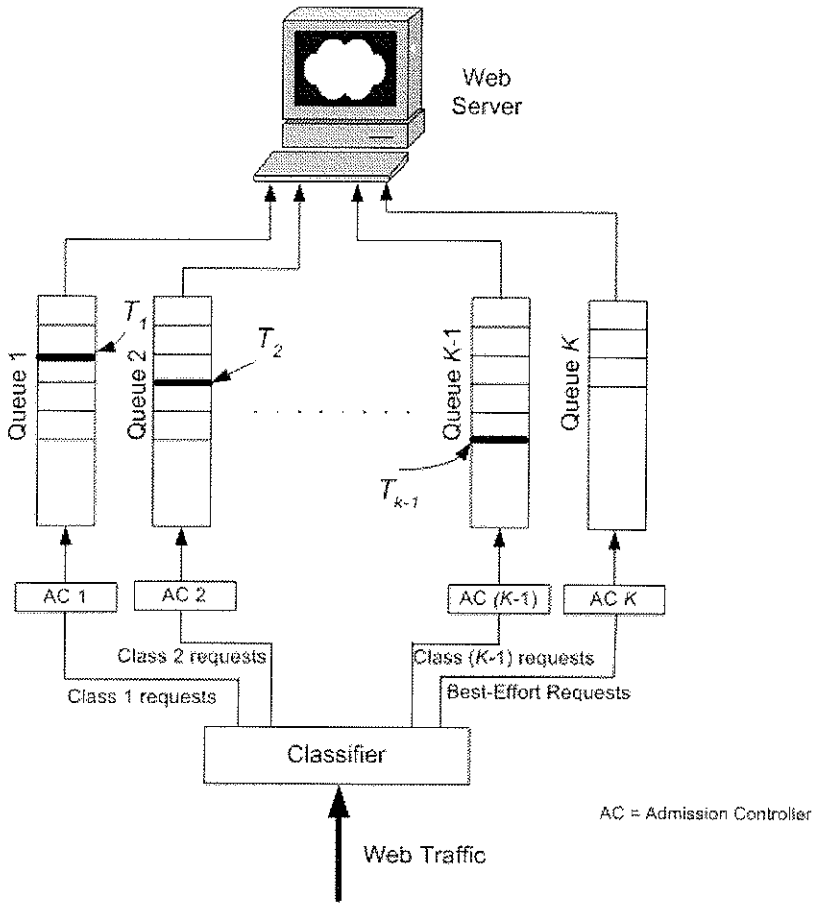

Fig. 4. Multiclass QoS-aware web server model

If the system is empty (the server in idle state) and a new request arrives, the server immediately switches to the corresponding queue and starts serving it. Now, let us assume that at some point in time the server is serving in any of the queue, say the ith queue, $1 \le i \le K$. Fig. 5 depicts the service policy in action in the server, assuming that the server is currently processing a class i request in the ith queue. While this service is ongoing, new requests may arrive in any of the queues and subsequently any of the QoS queues may reach or cross the corresponding threshold point regarding the number of queued up requests. When the service completes for the current request, two situations

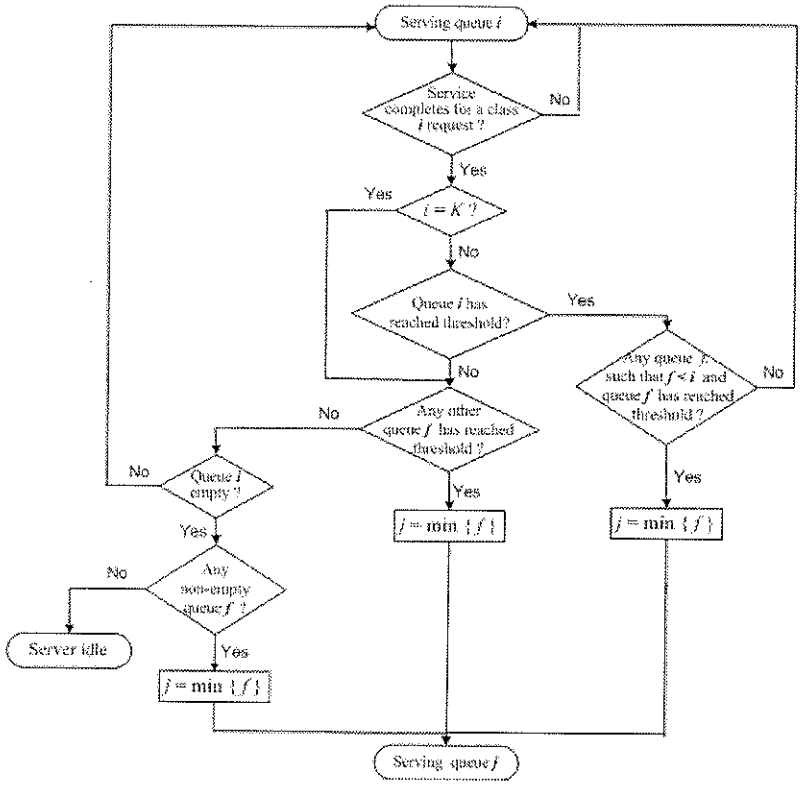

Fig. 5. Flow chart of the differentiated service in the server

may arise. In the first situation, the ith queue might have reached its threshold point (Note that the best-effort queue, i.e. for $i = K$, will not be in such situation because it does not have any threshold point set). The server will continue to serve in the ith queue only if none of the higher priority queues (if any) has reached their respective threshold points. If only one of these higher priority queues has reached its threshold, the server will switch to that queue. But, if there are two or more such queues, the server will switch to the highest priority one among them. In the other situation, the ith queue has not reached its threshold point or the service completion occurred in the best-effort queue. The server will remain in the current queue, only if the condition holds that none of the QoS queues has reached the respective threshold point and the current queue has not

become empty. As before, if there is only one threshold reaching QoS queue the server will switch to that queue and in case, there are more than one, the server will switch to the highest priority one of them. However, if the ith queue has become empty, even if none of the QoS queues has reached threshold the server will switch to the highest priority non-empty queue (if all other queues are empty the server will enter into the idle state).

5.2 Objective Formulation

Given the above service policy, the thresholds set at the QoS queues constitute the control parameters. In other words, for each of the QoS classes, the service provider has a control parameter, which he needs to set to appropriate values. To accomplish this, we have to express the performance measures explicitly in terms of the thresholds $[T_1, \cdots, T_{K-1}]$. These thresholds can only be set to those values, where the resulting performance measures conform to the performance guarantees. For meeting the optimization objective, these guarantees are treated as constraints and the thresholds are tuned to optimize the objective function while meeting these constraints.

5.3 Markov Chain Analysis

Let $N_i(t)$ be the number of class i requests in the system at time t, $I(t)$ be the queue which is being served, $J(t)$ be the service phase of the request currently being served,

where $i = 1, \cdots, K$ and $t \geq 0$. Denote by $N(t) = (N_1(t), \ldots, N_K(t))$. Define $X(t) = (N(t), I(t), J(t))$. Then it is easy to see that $X(t)$ is CTMC. Denote by $(n, i, j) = (n_1, \cdots, n_K, i, j)$ its generic state along with a server idle state $(\underbrace{0, \cdots, 0}_{K})$. The state space of the Markov chain is

$$\Delta = \{(\underbrace{0, 0, \cdots, 0}_{K})\} \cup$$

$$\{(n_1, \ldots, n_K, i, j) \mid n_1, \ldots, n_K \geq 0; 1 \leq i \leq \sum_{v=1}^{K} (1 - \delta_{0, n_v}); 1 \leq j \leq m; \{n_1, \ldots, n_K\} \neq \mathbf{0}\}$$

where δ_{0, n_v} is equal to 1 when $n_v = 0$, and is equal to 0 otherwise.

Let $Q_{(n, i, j) \to (n', i', j')}$ represent the transition rate of the Markov Chain $X(t)$ from state (n, i, j) to state (n', i', j'). All states are labeled in lexicographic order. Denote by Q the transition rate matrix of the Markov Chain $X(t)$. Besides we introduce some new matrices as follows. First, we focus on those state transitions, in which $N_1(t) = n_1$ and $N_1(t + \Delta t) = n_1'$, where $\Delta t \to 0$, i.e., we consider state transitions of the following form

$$(n_1, *, \cdots, *) \to (n_1', *', \cdots, *').$$

For fixed n_1 and n_1', the rates corresponding to these state transitions can be written out in a matrix form, which will be denoted by $A^{(1)}_{n_1 \to n_1'}$. More general, we define $A^{(g)}_{(n_1, \ldots, n_g) \to (n_1', \ldots, n_g')}$ $(g = 1, \ldots, K)$, which corresponds to the state transition of the following form:

$$(n_1, \cdots, n_g, *, \cdots, *) \to (n_1', \cdots, n_g', *', \cdots, *')$$

Applying the above notation, we can write out the transition rate matrix of Markov chain $X(t)$ as follows:

$$Q=\begin{pmatrix} A^{(1)}_{0\to 0} & A^{(1)}_{0\to 1} & & & & \\ A^{(1)}_{1\to 0} & A^{(1)}_{1\to 1} & A^{(1)}_{1\to 2} & & & \\ & A^{(1)}_{2\to 1} & A^{(1)}_{2\to 2} & A^{(1)}_{2\to 3} & & \\ & & A^{(1)}_{3\to 2} & A^{(1)}_{3\to 3} & A^{(1)}_{3\to 4} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}.$$

Obviously, $X(t)$ is still a Quasi-Birth-Death (QBD) Markov Chain. In further analysis, we shall see that it has the special level-dependence nature as was seen in single QoS class case in chapter 4.

5.4 Block Matrices Composing the Matrix Q

In what follows, we give a recursive formulae which allow us to obtain $A^{(g)}_{(n_1,\cdots,n_g)\to(n'_1,\cdots,n'_g)}$ from $A^{(g+1)}_{(n_1,\cdots,n_{g+1})\to(n'_1,\cdots,n'_{g+1})}$. Let $\boldsymbol{n}=(n_1,\cdots,n_g)$ and $\boldsymbol{n}'=(n'_1,\cdots,n'_g)$. Note that, all matrices for transitions where $\sum_{i=1}^{g}|n'_i-n_i|>1$ will contain only zeros due to the property of a CTMC. Therefore, in describing the recursion, we

explicitly show cases only where $\sum_{i=1}^{g} |n_i' - n_i| \in (0,1)$. The matrices of transitions of the form $A^{(K)}_{\boldsymbol{n}\to\boldsymbol{n}'}$ provide the end point of the recursion.

5.4.1 Recursive Formulae for $1 \le g < K$

First, consider the case, where a transition occurs because a request in queue f $(1 \le f < g)$, gets serviced.

Then, $\boldsymbol{n} = (n_1, \cdots, n_f, \cdots, n_g)$ and $\boldsymbol{n}' = (n_1, \cdots, n_{f-1}, n_f - 1, n_{f+1}, \cdots, n_g)$. Because there can be nor arrival or service completion in any other queue (because we are dealing with a CTMC), we have

$$A^{(g)}_{\boldsymbol{n}\to\boldsymbol{n}'} = \begin{pmatrix} A^{(g+1)}_{(\boldsymbol{n},0)\to(\boldsymbol{n}',0)} & & & \\ & A^{(g+1)}_{(\boldsymbol{n},1)\to(\boldsymbol{n}',1)} & & \\ & & A^{(g+1)}_{(\boldsymbol{n},2)\to(\boldsymbol{n}',2)} & \\ & & & \ddots \end{pmatrix}$$

Next, consider the case where an arrival occurs of a class f $(1 \le f < g)$ request.

Then, $\boldsymbol{n} = (n_1, \cdots, n_f, \cdots, n_g)$ and $\boldsymbol{n}' = (n_1, \cdots, n_{f-1}, n_f + 1, n_{f+1}, \cdots, n_g)$. Following the same argument as in the previous paragraph, we can write

$$A^{(g)}_{\boldsymbol{n}\to\boldsymbol{n}'} = \begin{pmatrix} A^{(g+1)}_{(\boldsymbol{n},0)\to(\boldsymbol{n}',0)} & & & \\ & A^{(g+1)}_{(\boldsymbol{n},1)\to(\boldsymbol{n}',1)} & & \\ & & A^{(g+1)}_{(\boldsymbol{n},2)\to(\boldsymbol{n}',2)} & \\ & & & \ddots \end{pmatrix}$$

Now, consider the case when there is no arrival or service completion for any of the class g requests. Then $\boldsymbol{n}' = \boldsymbol{n} = (n_1, \cdots, n_g)$. Because there is no arrival or service completion in the g th queue, we need to consider arrival or service completion in any of the remaining queues. Consequently,

$$A^{(g)}_{\boldsymbol{n}\to\boldsymbol{n}} = \begin{pmatrix} A^{(g+1)}_{(\boldsymbol{n},0)\to(\boldsymbol{n},0)} & A^{(g+1)}_{(\boldsymbol{n},0)\to(\boldsymbol{n},1)} & & \\ A^{(g+1)}_{(\boldsymbol{n},1)\to(\boldsymbol{n},0)} & A^{(g+1)}_{(\boldsymbol{n},1)\to(\boldsymbol{n},1)} & A^{(g+1)}_{(\boldsymbol{n},1)\to(\boldsymbol{n},2)} & \\ & A^{(g+1)}_{(\boldsymbol{n},2)\to(\boldsymbol{n},1)} & A^{(g+1)}_{(\boldsymbol{n},2)\to(\boldsymbol{n},2)} & A^{(g+1)}_{(\boldsymbol{n},2)\to(\boldsymbol{n},3)} \\ & \ddots & \ddots & \ddots \end{pmatrix}$$

5.4.2 Transition Matrices $A^{(K)}_{\boldsymbol{n}\to\boldsymbol{n}'}$

Now, we find the matrices that will provide the end point of the recursive formulae. Now, we find the matrices for transitions $A^{(K)}_{\boldsymbol{n}\to\boldsymbol{n}'}$. Define y, z and ψ_v as:

y = number of nonzero entries in $\boldsymbol{n}$

z = number of nonzero entries in $\boldsymbol{n}'$

$$\psi_v = \sum_{\sigma=1}^{v} \delta_{0,n_\sigma}$$

where δ_{0,n_σ} is equal to 1 when $n_\sigma = 0$, and is equal to 0 otherwise

For the following analysis, we define a few matrices.

$\Gamma_{p:q}^{u\to v}$ is a matrix such that

$$\Gamma_{p:q}^{u\to v}(i,j) = \begin{cases} 1, & \text{if } i = u \text{ and } j = v \\ 0, & \text{otherwise} \end{cases}$$

where $1 \le i \le p,\ 1 \le j \le q$. In words, $\Gamma_{p:q}^{u\to v}$ is a $p \times q$ matrix which has 1 in the cell at the intersection of the row u and column v, with all other entries being 0.

$\mathbf{0}_{c\times d}$ is a $c \times d$ dimension matrix with 0 in all entries and $I(c)$ is an identity matrix of dimension $c \times c$.

ς_p is a square matrix such that

$$\varsigma_p\ (i,j) = \begin{cases} S_i - \lambda I(m), & \text{if } i = j \\ \mathbf{0}_{m\times m}, & \text{otherwise} \end{cases}$$

where $1 \le i \le p,\ \lambda = \lambda_1 + \cdots + \lambda_K$. The description of the matrix is in terms of constituent sub matrices, which are of dimension $m \times m$. The sub matrices at the diagonals contain non-zero entries as shown and the other sub matrices are zeros.

$\Omega_{p:q}$ is a matrix such that

$$\Omega_{p:q}(i,j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

where $1 \le i \le p,\ 1 \le j \le q$. This is, in fact, a matrix of the form $\left[I(p)\ \mathbf{0}_{p\times(q-p)} \right]$.

As before, we first consider the situation, where a transition occurs because a request in queue $f\ (1 \le f \le K)$ gets serviced. Then, $\boldsymbol{n} = (n_1, \cdots, n_f, \cdots, n_K)$ and $\boldsymbol{n'} = (n_1, \cdots, n_{f-1}, n_f - 1, n_{f+1}, \cdots, n_K)$. Following cases may occur after the transition and we deal with them individually.

Case 1: The number of requests in queue f has reached threshold i.e. $n_f > T_f$. Two situations may arise as follows:

- No higher priority queue has reached its threshold i.e. $\{\sigma \mid 1 \le \sigma < f; n_\sigma \ge T_\sigma\} = \varnothing$. As a result, the server stays in queue f. Therefore,

$$A_{\boldsymbol{n}\to\boldsymbol{n'}}^{(K)} = \Gamma_{y:z}^{(f-\psi_f)\to(f-\psi_f)} \otimes (S_f^0 \beta_f)$$

- At least one higher priority queue has reached threshold i.e. $\{\sigma \mid 1 \le \sigma < f; n_\sigma \ge T_\sigma\} \ne \varnothing$. Therefore, the server will switch to the highest priority one of them. Therefore,

$$A_{\boldsymbol{n}\to\boldsymbol{n'}}^{(K)} = \Gamma_{y:z}^{(f-\psi_f)\to(h-\psi_h)} \otimes (S_f^0 \beta_h),$$

where $h = \min\{\sigma \mid 1 \le \sigma < f; n_\sigma \ge T_\sigma\}$

Case 2: The threshold in queue f has not been reached i.e. $n_f \le T_f$. Following situation may be true in such case.

- None of the other queues have reached threshold i.e. $\{\sigma \mid 1 \le \sigma \le K; \sigma \ne f; n_\sigma \ge T_\sigma\} = \varnothing$. However, there could be two possibilities.
  - Queue f has one or more requests waiting i.e. $n_f > 1$. Since no queue has reached threshold, the server stays in queue f. Consequently,

$$A_{\boldsymbol{n}\to\boldsymbol{n}'}^{(K)} = \Gamma_{y:z}^{(f-\psi_f)\to(f-\psi_f)} \otimes (S_f^0 \beta_f)$$

- Queue f becomes empty after the service i.e. $n_f = 1$. There could be two possibilities.
  - There is at least one non-empty queue i.e. $\{\sigma \mid 1 \le \sigma \le K; \sigma \ne f; n_\sigma \ge 0\} \ne \varnothing$. Therefore, the server switches to the highest priority non-empty queue. Consequently,

$$A_{\boldsymbol{n}\to\boldsymbol{n}'}^{(K)} = \Gamma_{y:z}^{(f-\psi_f)\to(h'-\psi_{h'})} \otimes (S_f^0 \beta_{h'}),$$

    where $h' = \min\{\sigma \mid 1 \le \sigma \le K; \sigma \ne f; n_\sigma \ge 0\}$.

  - All other queues are empty i.e. $\{\sigma \mid 1 \le \sigma \le K; \sigma \ne f; n_\sigma \ge 0\} = \varnothing$. Therefore, the server returns to idle state.

$$A_{\boldsymbol{n}\to\boldsymbol{n}'}^{(K)} = S_f^0$$

- At least one of the other queues has reached threshold i.e. $\{\sigma \mid 1 \le \sigma \le K; \sigma \ne f; n_\sigma \ge T_\sigma\} \ne \varnothing$ Therefore,

$$A_{\boldsymbol{n}\to\boldsymbol{n}'}^{(K)} = \Gamma_{y:z}^{(f-\psi_f)\to(h''-\psi_{h''})} \otimes (S_f^0 \beta_{h''}),$$

where $h'' = \min\{\sigma \mid 1 \le \sigma \le K; \sigma \ne f; n_\sigma \ge T_\sigma\}$.

We now consider the situation, where a transition occurs because a request arrives in queue f $(1 \le f \le K)$. Then, $\boldsymbol{n} = (n_1, \cdots, n_f, \cdots, n_K)$ and $\boldsymbol{n}' = (n_1, \cdots, n_{f-1}, n_f + 1, n_{f+1}, \cdots, n_K)$. If all the queues have been empty, the server will immediately switch to the queue f to serve the newly arrived request. But, if the server

has been serving any request when the arrival occurs, it will stay continue serving that request. Consequently,

$$A^{(K)}_{\boldsymbol{n}\to\boldsymbol{n}'} = \begin{cases} \lambda_f \beta_f, & \text{if } \boldsymbol{n} = \mathbf{0}_{1\times K} \\ \Omega_{y:z} \otimes \lambda_f I(m), & \text{otherwise} \end{cases}$$

We now consider the situation, where no transition occurs because no arrival or service completion occurs in any of the queues. Therefore, $\boldsymbol{n}' = \boldsymbol{n} = (n_1, \cdots, n_f, \cdots, n_K)$. It is the situation where the state of the system does not change. Consequently,

$$A^{(K)}_{\boldsymbol{n}\to\boldsymbol{n}} = \begin{cases} -\lambda, & \text{if } \boldsymbol{n} = \mathbf{0}_{1\times K} \\ \Gamma^{(f-\psi_f)\to(h''-\psi_{h''})}_{y:z} \otimes \varsigma_y, & \text{otherwise} \end{cases}$$

where $\lambda = \lambda_1 + \cdots + \lambda_K$.

5.5 Steady State Distributions

If we closely look at the block matrices for different levels in the Markov chain, we will find some interesting observations. First, consider the matrices $A^{(1)}_{n_1\to n_1'}$, where $n_1' = n_1 + 1$, for $n_1 \geq 1$. Because, a transition has taken place in the topmost layer of the Markov chain, due to the property of CTMC, no transition can take place in the lower layers. As a result, the submatrices covering these lower layers will be exactly same to each other, when transitions of same behavior occur in the topmost layer. However, we need to consider the transition matrix $A^{(1)}_{0\to 1}$ differently from these matrices, because we

have to take into account the possibility of queue 1 being empty. Consequently, we can write

$$A^{(1)}_{1\to 2} = A^{(1)}_{2\to 3} = \cdots$$

Following a similar argument, we have

$$A^{(1)}_{2\to 1} = A^{(1)}_{3\to 2} = \cdots$$

Now, consider the transition matrices $A^{(1)}_{n_1\to n_1}$, for $n_1 < T_1$. Because no transition has occurred in the topmost layer, the remaining lower layers will characterize these matrices. Every possible transition in these lower layers will take place with the same assumption that queue 1 has not reached threshold.

Consequently, these matrices will be equal to each other, and we can write

$$A^{(1)}_{1\to 1} = A^{(1)}_{2\to 2} = \cdots = A^{(1)}_{T_1-1\to T_1-1}$$

Following a similar argument with the assumption that queue 1 has reached threshold, when $n_1 \geq T_1$, we can write

$$A^{(1)}_{T_1\to T_1} = A^{(1)}_{T_1+1\to T_1+1} = \cdots$$

As a result, the matrix Q rewritten below assumes the same level dependent structure as was seen in chapter 4 with the two-queue model:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ T_1 - 1 \\ T_1 \\ \vdots \end{array} \begin{pmatrix} A^{(1)}_{0\to 0} & A^{(1)}_{0\to 1} & & & & & & & \\ A^{(1)}_{1\to 0} & A^{(1)}_{1\to 1} & A^{(1)}_{1\to 2} & & & & & & \\ & A^{(1)}_{2\to 1} & A^{(1)}_{1\to 1} & A^{(1)}_{1\to 2} & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & & & A^{(1)}_{2\to 1} & A^{(1)}_{1\to 1} & A^{(1)}_{1\to 2} & & \\ & & & & & A^{(1)}_{2\to 1} & A^{(1)}_{T_1\to T_1} & A^{(1)}_{1\to 2} & \\ & & & & & & A^{(1)}_{2\to 1} & A^{(1)}_{T_1\to T_1} & A^{(1)}_{1\to 2} \\ & & & & & & \ddots & \ddots & \ddots \end{pmatrix}$$

Therefore, it is obvious that we can follow the same solution procedure for steady state distributions as was done in chapter 4. However, since we allow infinite capacity in all of the queues, there will be truncations involved in calculating the R matrices and intermediate submatrices.

5.6 Computing Performance Measures and Setting the Control Parameters

Denote by x the steady-state distribution of Markov chain $X(t)$. Corresponding to the construction of matrix Q, we partition x into subvectors and assume that $x = [x_0, x_1, x_2, \cdots]$, where x_{n_1} is partitioned into smaller subvectors

$$x_{n_1} = [x_{(n_1,0)}, x_{(n_1,1)}, x_{(n_1,2)}, \cdots], \quad n_1 \geq 0,$$

and also $x_{(n_1,n_2)}$ can be further partitioned as

$$x_{(n_1,n_2)} = [x_{(n_1,n_2,0)}, x_{(n_1,n_2,1)}, x_{(n_1,n_2,2)}, \cdots].$$

More general, we can write

$$x_{(n_1,n_2,\cdots,n_g)} = [x_{(n_1,n_2,\cdots,n_g,0)}, x_{(n_1,n_2,\cdots,n_g,1)}, x_{(n_1,n_2,\cdots,n_g,2)}, \cdots]$$
$$1 \leq g \leq K-1; n_1, n_2, \cdots, n_g \geq 0$$

Finally, all of these vectors are partitioned into basic subvectors $x_{(n_1,n_2,\cdots,n_K)}$ $(n_1, n_2, \cdots, n_K \geq 0)$.

Now, it is quite straight forward to obtain joint probabilities of steady state queue lengths from the steady state distributions of $X(t)$:

$$\text{Prob}\{N_1 = n_1, N_2 = n_2, \cdots, N_K = n_K\} = \boldsymbol{x}_{(n_1, n_2, \cdots, n_K)}\, e_{x\,(n_1, n_2, \cdots, n_K)},$$
$$\forall\, n_1, n_2, \cdots, n_K = 0, 1, 2, \cdots$$

where N_f represents the queue length of class f requests in the system when it is in steady-state, and $e_{x\,(n_1, n_2, \cdots, n_K)}$ represents a column vector in which all elements are 1.

Let $q_i(f)$ be the steady state probability that there are i requests in queue f. Therefore,

$$q_i(f) = \sum_{n_1=0}^{\infty} \cdots \sum_{n_{f-1}=0}^{\infty} \sum_{n_f=i}^{i} \sum_{n_{f+1}=0}^{\infty} \cdots \sum_{n_K=0}^{\infty} \boldsymbol{x}_{(n_1, n_2, \cdots, n_K)}\, e_{x\,(n_1, n_2, \cdots, n_K)}.$$

The mean queue length for class f requests is then obtained as follows:

$$L_f = \sum_{i=1}^{\infty} i q_i(f).$$

The mean response times $\overline{\tau}_f\,(1 \le f \le K)$ are straightforward to obtain by using Little's law:

$$\overline{\tau}_f = \frac{L(f)}{\lambda_f}, \qquad 1 \le f \le K.$$

Once the relationships between the performance measures and the threshold points are found out using the above procedure, finding the appropriate values for these control parameters is also possible. The relationships can be fitted to appropriate functions by using regression analysis. Now, the service provider may want to reach an optimization objective in addition to meeting the performance guarantees and have the optimization functions defined in terms of the performance measures. Because the performance measures are explicitly expressed in terms of the control parameters by our analytic

method, the objective functions can now be defined in terms of these parameters as well and some non-linear constrained optimization techniques can be applied to find out their optimal values.

Let us consider the optimization objective to be the one in which we want to minimize the mean response time of the best-effort requests, while maintaining the guaranteed mean response times in the QoS queues, we can write the optimization problem formally as follows.

$$\begin{aligned} \text{minimize} \quad & \bar{\tau}_k(T_1, T_2, \cdots, T_{K-1}) \\ \textit{subject to} \quad & \bar{\tau}_1(T_1, T_2, \cdots, T_{K-1}) \leq \delta_1 \\ & \bar{\tau}_2(T_1, T_2, \cdots, T_{K-1}) \leq \delta_2 \\ & \vdots \\ & \bar{\tau}_{K-1}(T_1, T_2, \cdots, T_{K-1}) \leq \delta_{K-1} \\ & T_1, T_2, \cdots, T_{K-1} \geq 0 \end{aligned}$$

where $\bar{\tau}_i(T_1, \cdots, T_{K-1})$, stands for the mean response time of the requests served in the ith queue, δ_i is the corresponding guaranteed mean response time and the $T_1, \cdots, T_{K-1}$ inside the bracket indicates that it is a function of these threshold points. The functions are likely to be non-linear ones and can be fitted by regression from sample data collected from numerical results from our analysis. Then, non-linear constrained optimization techniques can be applied to find the sought optimum points [Murty95]. Commercially available optimization softwares such as LINDO [LINDO] can now do such optimizations with possibly thousands of parameters.

However, the objective can be varied according to the policy of the service provider and pricing policies can be in place as well. We argue that our methodology can be applicable to those scenarios as well, because we know the direct relationship between the performance measures and the control parameters from our analysis

6 Results and Discussion

For demonstrating the applicability of our analysis in a practical scenario, we conducted our analytical method using a real web trace [EPHttp]. We then validated the numerical results by performing simulation on the same trace data. In the following sections we describe the methodology of our analysis, results and simulation experiments.

6.1 Computational Methodology and Related Assumptions

All the steps described in our models were programmed in Matlab [MatLa], which offers efficient matrix manipulation tools required for our analysis. Once the analytical model is programmed, we feed it with the requests from a processed trace file. We use the EPA-HTTP trace [EPhttp] that contains a day's worth of all HTTP requests to the EPA WWW server located at Research Triangle Park, NC. From the trace file, we collected information on request arrival times and requested file sizes in each of the requests. In the trace, the requests are logged in an ASCII format, one request per line with information on the source IP, arrival time, requested url, HTTP reply code and the number of bytes delivered for the url request.

We have seen in Chapter 2, that there can be numerous criteria on which the requests in a web server can be classified. Available web traces, however, do not contain any information on the classification of the recorded web traffic. In our analysis, in the absence of any indication of such classification in the trace, categorizing the requests was

done by randomly assigning the requests to two classes, following some arrival rate ratio for them. Class 1 is the QoS class and class 2 is the best-effort. The inter-arrival times were fitted to the exponential distribution with mean arrival rates per second $\lambda_1 = 3$ and $\lambda_2 = 2$ where λ_1 and λ_2 are the corresponding arrival rates of classes 1 and 2 respectively. The queue limit for best-effort requests was set to 25.

For the service time distribution, we collected the file size information of the requested urls and adopted a service processing speed of 400 *kbps*. Note that, the selection of the processing speed depends on the server characteristics and configurations, which can vary widely. Experimenting with the actual serving times for different requests in the actual server may also provide us with the relationship between the characteristics (such as file size) of the requested files and their processing times. For our analysis, we take the above processing speed for demonstration purpose. In a practical scenario, we assume however that, this preprocessing will be done as a separate step before our model is fed with the data. The resulting service time values were then fitted to a continuous phase-type distribution. We used the EMpht software [EMSoft] to perform the fitting of the phase-type distribution. The software implements the method developed by Asmussen, Nerman, and M. Olsson [AsNe96] using EM-algorithm. Fig. 6 shows the cumulative distribution functions for the input and the fitted distribution of the service times. We see that the EM-algorithm method gives a very nice close fit for our input traces.

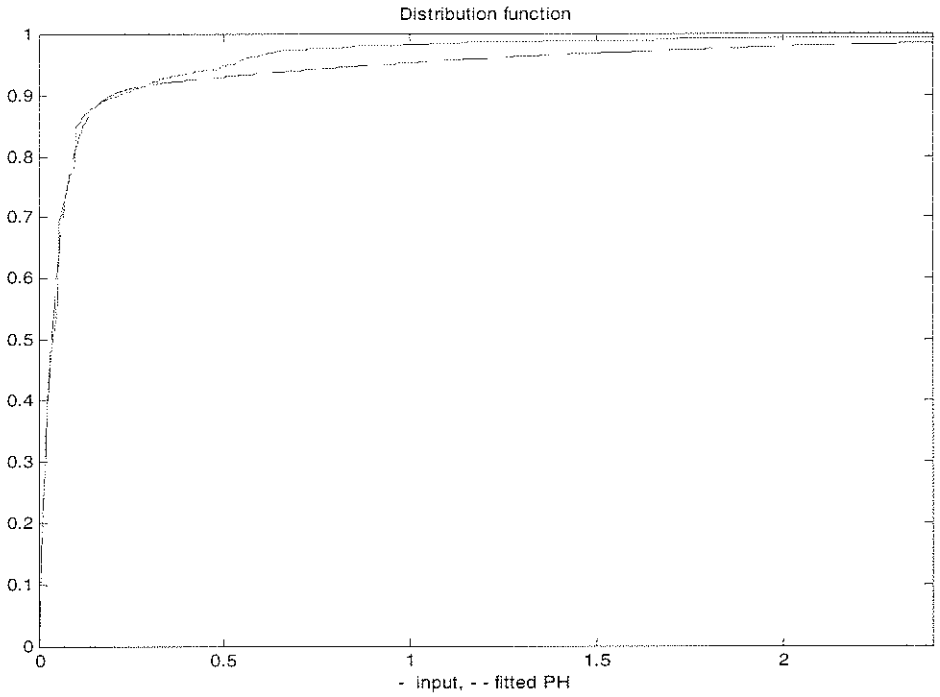

Fig. 6. CDF of service times of requests in the trace fitted to phase-type

6.2 Numerical Results

Fig. 7 shows the relationship for the QoS traffic between the average response time and the values of the threshold point. As seen from the figure, the mean response time for the QoS requests rises with the increasing values of the threshold T. If the threshold value is set to a low value, the server will switch to the QoS queue when only a small number of QoS requests are queued up. Once the server switches to the QoS queue, it serves exhaustively. As a result, lower value of threshold will reduce the overall waiting time of the QoS requests. Conversely, the average waiting time of the QoS requests will

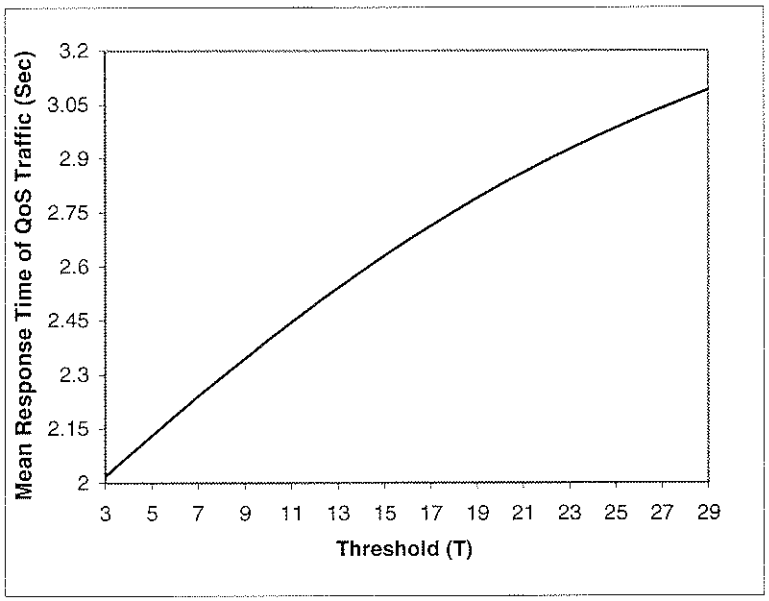

Fig. 7. Mean response time of QoS traffic vs. threshold in QoS queue

be higher, when threshold point is set to higher values. This explains the steady increase of mean response time for QoS requests with the increasing threshold.

Now, we see the behavior of the response time for the best-effort traffic in Fig. 6 and find an interesting trend. The mean response time for the best-effort requests starts decreasing with the increasing threshold set at the QoS queue. But, after the threshold reaches a certain value, the mean response time starts rising with the increasing threshold values. The rising trend persists thereafter. When the threshold in the QoS queue is very low, the best-effort queue will have less opportunity to get served, because even a small number of QoS requests being queued up will make the server switch from the best-effort queue. Setting threshold to a higher value should increase the amount of time the best-effort queue gets from the server and reduce the mean response time. However, we see that this

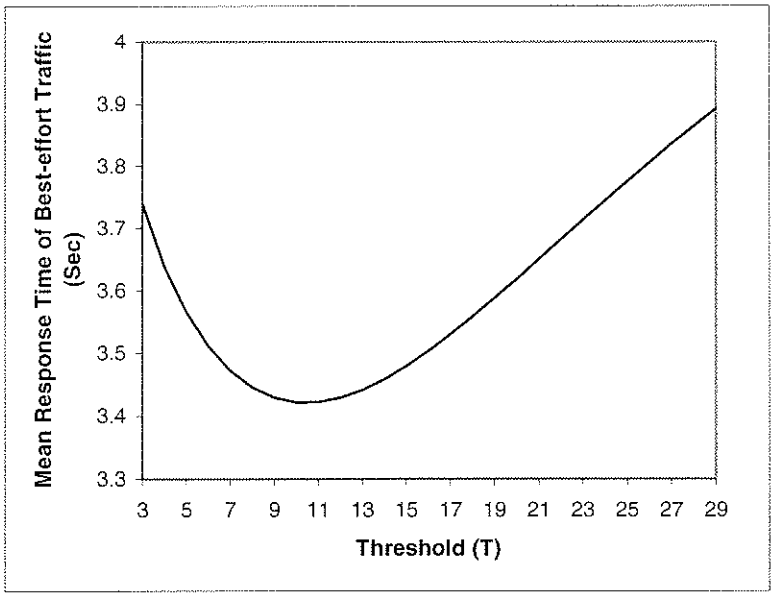

Fig. 8. Mean response time of best-effort traffic vs. threshold in QoS queue

remains true only until some value of the threshold, beyond which the response time increases if the threshold increases. We can explain this phenomenon if we notice that, once the server switches to the QoS queue it serves it until that queue becomes empty again. We see that, if the threshold point is reached in the QoS queue while the server is currently serving in the best-effort queue, the server switches to the QoS queue after finishing the current request in the best-effort queue. If at that point, the number of requests queued up in the QoS queue is higher, the server will take a longer time to finish them first before switching to the best-effort queue again. Higher threshold obviously makes it more likely to happen. As a result, the mean response time for the best-effort queue rises after this particular threshold point.

Fig. 8 shows the loss probability of the best-effort requests with varying threshold points. It follows the same trend as the mean response time in the best-effort queue. It

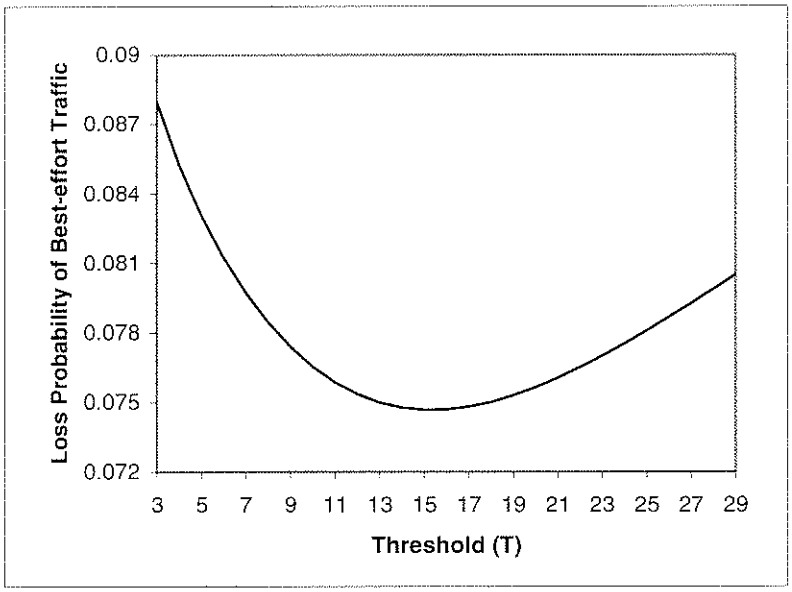

Fig. 9. Loss Probability of best-effort traffic vs. threshold in QoS queue

starts to decrease at the beginning, but continues to rise after some particular threshold point. Note that, the loss probability is here the dropping probability after a best-effort request entered the queue after being admitted, but was dropped because of the limited capacity of the best-effort queue.

6.3 Achieving Optimization Objective: An Example

Given the above numerical results obtained from the analysis, let's assume that the mean response time for QoS requests is guaranteed to be at most 2.75 *seconds*. Also assume that the service provider determines his optimization objective to be the minimization of the mean response time of the best-effort requests, while maintaining this guarantee for the QoS requests. We can formulate this optimization objective in the following way:

$$\begin{aligned} &\text{minimize} && \bar{\tau}_2(T) \\ &\textit{subject to} && \bar{\tau}_1(T) \le 2.75\,\text{sec} \\ & && T \ge 0 \end{aligned}$$

We fit the resulting sample points of mean response times from our analysis to two suitably selected non-linear functions giving acceptably good fitting, one for each of the queues. We used the DataFit software [Dataf], which finds the best fits for the data from nearly hundreds of functions. Now, we can rewrite the optimization objective with the fitted functions:

$$\begin{aligned} &\text{minimize} && (-1.115\times10^{-04}T^3 + (7.165\times10^{-3}T^2 - 0.118T + 4.002) \\ &\textit{subject to} && (-7.007\times10^{-4}T^2 + 6.376\times10^{-2}T + 1.83) \le 2.75 \\ & && T \ge 0 \end{aligned}$$

We then solve the above optimization problem using the LINDO [LINDO] optimization software and get the optimal value for threshold T_1 to be 10. This optimal value of control parameter T_1 is validated also from the plots of the numerical results in Fig. 7-8.

6.4 Simulation Results

To validate the analytical model and its results we also ran some simulation experiments using a discrete-event simulator Parsec [BaMe98].

6.4.1 Simulation Setup

We implemented the service policy described in the thesis, in a simulated web server. A classifier was implemented at the front end interface of the web server. Two separate FIFO queues were implemented to hold the classified requests. Each of the incoming requests was tagged with its arrival time and request size information. We generated the requests from the trace file following the Poisson arrival process with the similar rates as those used in the numerical analysis. The classifier took the incoming requests and then redirected them into two separate queues randomly. QoS requests were directed to queue 1 and the best-effort requests were directed to queue 2. The random selection process is, however, consistent with the relative arrival rates of the two classes of requests and with that in the numerical analysis. The second queue has the same capacity as the one used in the numerical analysis. When that queue becomes full, subsequent requests are rejected. The storage entity in the simulated web server is implemented as a disk which has a linear content retrieval time i.e. reading a file from the disk will take time which is proportional to the stored file size. The time required to process the request is also proportional to the size of the file requested. The retrieval time and this processing time sum to the service time. As a result, the service time is also proportional to the request size. The speed of the service is kept equal to the value used in the numerical analysis. After the service time elapses, it is tagged with the finish time and delivered from the server entity to an analyze entity. The analyzer entity determines the total response time for the request by deducting the arrival time from the service finish time. After all the

requests in the trace file are processed, mean response times are calculated by taking the average.

6.4.2 Results

We conducted a number of simulation runs with different values of threshold for QoS requests. Fig. 10-12 show the simulation results by dotted lines beside the numerical results from our analytical model in solid lines. We can see that the simulation results follow the analytical results very closely and validate the numerical results.

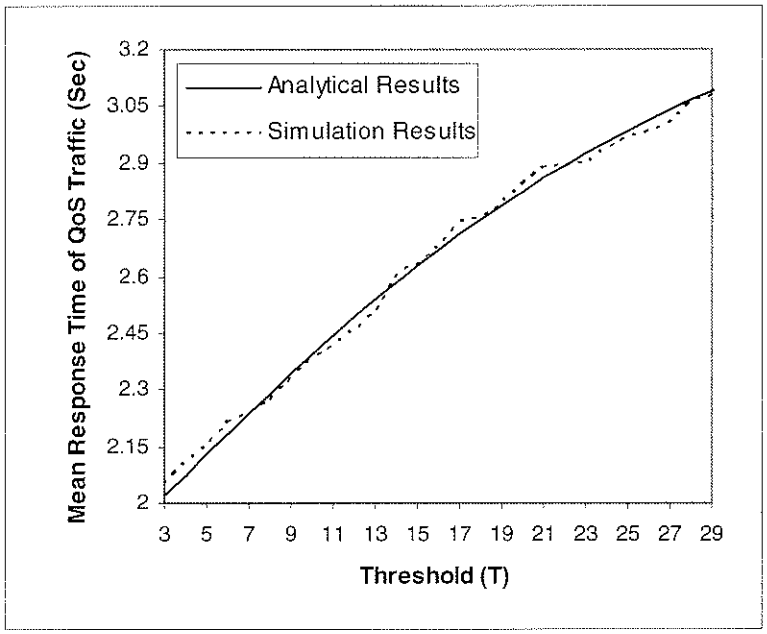

Fig. 10. Comparison of analytical and simulation results for response times of QoS traffic

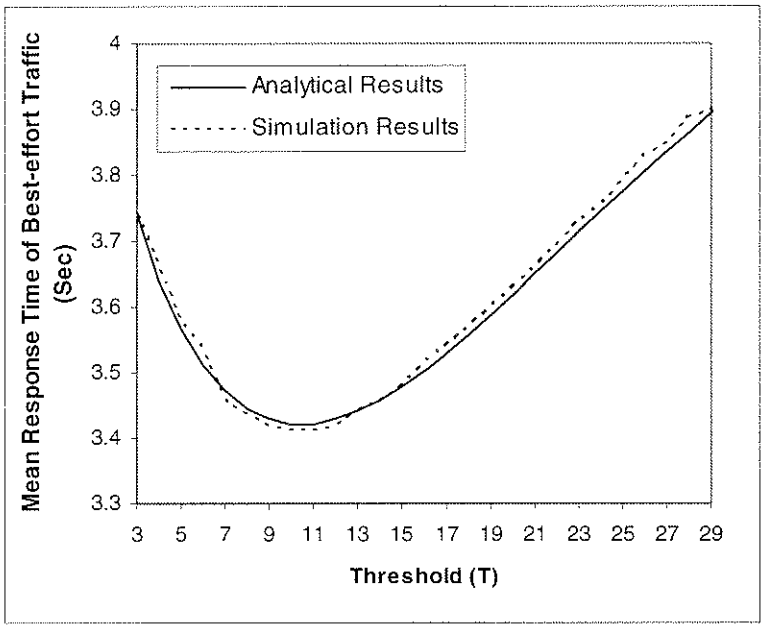

Fig. 11. Comparison of analytical and simulation results for response times of best-effort traffic

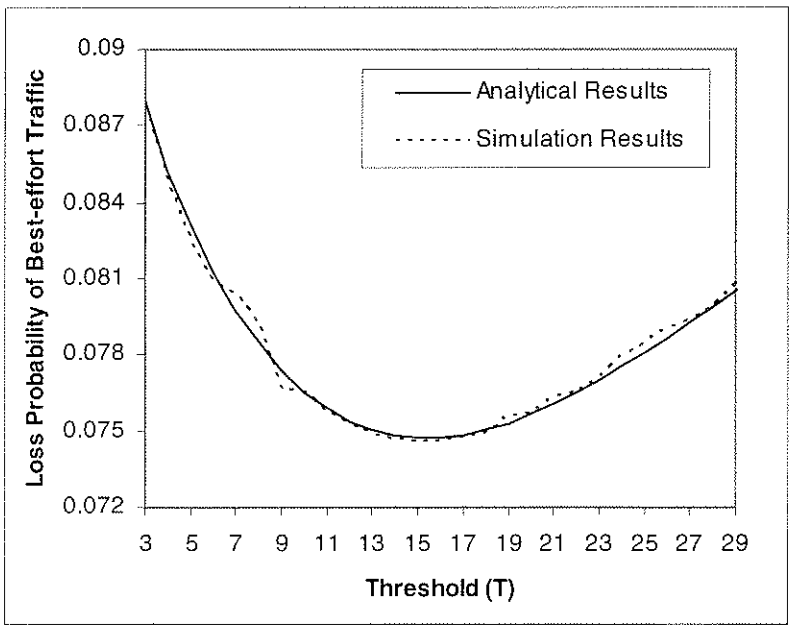

Fig. 12. Comparison of analytical and simulation results for loss probability in best-effort traffic

7 Conclusions and Future Work

In this chapter, we summarize the works presented in this thesis, major contributions of this thesis and some possible future directions.

We consider QoS aware web server and devise an analytic method to obtain a controllable service differentiation policy for it. The major contributions of this thesis are listed below:

- We proposed a differentiated service policy in web servers to control the quality of service provided to different classes of requests. The policy is easily deployable in existing web servers and provides great amount of control to the operator. The web server is thus QoS-aware and can allocate its resources according to the QoS requirements of different classes of web traffic it handles.
- We solved the resulting system model using a unique queueing theoretic approach and derived expressions for the performance measures in terms of the control parameters used in the differentiated service policy. Since the performance measures are expressed explicitly in terms of these parameters, optimization can be done using traditional optimization methods once a particular optimization objective is given.
- The system model developed for the multiclass scenario needs us to solve the multiclass version of the threshold-based polling scheme as known in queueing theory literature. An important contribution of this thesis is the solution of this queueing problem without assuming exponential service time distribution. No

known solution exists which considers arbitrary service time distribution other than exponential. The assumption of exponential service time distribution in web servers is impractical and will give erroneous results in the analysis. We consider phase-type distribution which can virtually model any practical service time distribution. As a result, our solution becomes the only known solution to the problem that can handle arbitrary service time distribution by the abstraction through phase-type distribution. We developed the solution using the matrix-geometric method, which means it is free from disadvantages of transform based methods [AlLi03]. Even the two-queue model of the threshold-based polling scheme considering phase-type service time distribution has no known solution for performance analysis other than the solution presented in this thesis.

7.1 Future Work

This research has opened some interesting directions for further research. Some of them are listed below.

- One of the improvements of the system model in this thesis would be to consider arrival processes which are not Poisson. Markovian Arrival Process (MAP) will be more practical as the arrival process and can be considered in our future work.
- In our present analysis, we do not consider caching in the web server. But, most of the practical web servers implement caching and need to be considered in the model. In our future works, we aim to include the effect of caching and different caching schemes on the performance measures as well.

- We only consider a single web server in this thesis. Many web hosting facilities implement clustered web servers to enhance performances. Though the single server considered in the thesis can be thought of as an abstraction of the clustered collection of servers, the more realistic analysis will be to consider the presence of the multiple servers and the switching delays and overheads among them.

Acronyms

CTMC	Continuous Time Markov Chain
IP	Internet Protocol
MAP	Markovian Arrival Process
QBD	Quasi-Birth-Death
QoS	Quality of Service
SLA	Service Level Agreement

References

[AbSh02] T. Abdelzaher, K. G. Shin, and N. Bhatti. "Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, Jan. 2002.

[AbSh99] T. F. Abdelzaher and K. G. Shin, "QoS Provisioning with qContracts in Web and Multimedia Servers," *Proc. IEEE Real-Time Systems Symposium*, Phoenix, AZ, Dec. 1999.

[AbSt01] C. Lu, T. Abdelzaher, J. Stankovic, and S. Son, "A feedback control approach for guaranteeing relative delays in Web servers," *Proc. IEEE Real-Time Technology and Applications Symp.*, Taipei, Taiwan, pp. 51-62, June 2001

[Alfa98] A. S. Alfa, "Matrix Geometric Solution of Discrete Time MAP/PH/1 Priority Queue," *Naval Res. Logistics*, vol. 45, pp. 23–50, 1998.

[AlLi03] A.S. Alfa, B. Liu, and Q.M.He, "Discrete-Time analysis of MAP/Ph/1 multiclass general preemptive queue," *Naval Research Logistics*, vol. 50, no. 6, pp. 662-682, 2003.

[AlMa98] J.Almedia, M.Dabu, A.Manikntty, and P.Cao, "Providing Differentiated Levels of Service in Web Content Hosting," *Proc. First Workshop on Internet Server Performance*, Madison, WI, June 1998.

[AsNe96] S. Asmussen, O. Nerman, and M. Olsson, "Fitting phase-type distributions via the EM algorithm," *Scand. J. Statist.*, vol. 23, pp. 419--441, 1996.

[BaCr99] P. Barford and M. Crovella, "A performance evaluation of hyper text transfer protocols," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 1, pp. 188-197, 1999.

[BaMe98] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B.Park, and H. Song, ``Parsec: A Parallel Simulation Environment for Complex Systems," *IEEE Computer*, vol. 31, no. 10, pp. 77-85, Oct. 1998.

[BhFr99] N. Bhatti and R. Friedrich, "Web Server Support for Tiered Services," *IEEE Network*, vol.13, no. 5, pp. 64-71, Sept. 1999.

[BoDo97] O.J. Boxma and D.G. Down, "Dynamic Server Assignment in a Two-queue Model," *European Journal of Operations Research*, vol. 103, pp. 595-609, 1997.

[BoKo95] O.J. Boxma, G. Koole and I. Mitrani, "A two-queue polling model with a threshold service policy," *Proc. MASCOTS '95*, eds. P. Dowd and E. Geleneb, IEEE Computer Society Press, Los Alamitos (CA), pp. 84-89, 1995.

[BoKo95a] O.J. Boxma, G. Koole and I. Mitrani, "A two-queue polling model with threshold switching," *Quantitative methods in parallel systems*, pp. 129-140, 1995.

[ChMo02] H. Chen and P. Mohapatra, "Session-based overload control in QoS-aware Web servers," *Proc. IEEE INFOCOM 2002*, New York, June 2002.

[Dataf] DataFit Curve Fitting and Data Plotting Software, http://www.curvefitting.com/, June 2004

[DoBo95] D.G. Down, O.J. Boxma, "A two-queue polling model with threshold switching," *Proc. Nordic Teletraffic Seminar NTS-12*, Espoo, Finland, August 1995.

[DoRa99] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," *Proc. ACM SIGCOMM*, pp. 109-119, Aug. 1999.

[EgHe99] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Servers," *World Wide Journal*, vol. 2, no. 3, pp.133-142, August 1999.

[EMSoft] EMpht software http://www.maths.lth.se/matstat/staff/asmus/pspapers.html

[EPHttp] EPA-HTTP - A Day of HTTP Logs from the EPA WWW Server. Available at: http://ita.ee.lbl.gov/html/contrib/EPA-HTTP.html

[GaHa88] H.R. Gail, S.L. Hantler, and B.A. Taylor, "Analysis of a non-preemptive priority multiserver queue," *Advances in Applied Probability*, vol. 20, pp. 852-879, 1988.

[GoLu00] L. Golubchik and J.C.S. Lui, "A fast and accurate iterative solution of a multi-class threshold-based queueing system with hysteresis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 1, pp. 196-206, 2000.

[HaId94] B. Haverkort, H.P. Idzenga and B.G. Kim, "Performance evaluation of ATM cell scheduling policies using Stochastic Petri Nets," Technical Report, University of Twente, Tele-Informatics and Open Systems Group TIOS 94-19, 1994.

[KaKn00] V. Kanodia and E. Knightly, "Multi-class latency-bounded Web services," *Proc. IEEE/IFIP IWQoS 2000*, Pittsburgh, PA, June 2000.

[KaKn02] V. Kanodia and E. Knightly, "Ensuring Latency Targets in Multiclass Web Servers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 10, October 2002.

[Lee93] D-S. Lee, "A two-queue model with exhaustive and limited service disciplines," Report C & C Research Laboratories, NEC USA Inc., 1993.

[LeLu04] S.C.M. Lee, J.C.S. Lee, and D.K.Y. Yau, "A Proportional-Delay Diffserv-Enabled Web Server: Admission Control and Dynamic Adaptation," *IEEE Transaction on Parallel and Distributed Systems*, vol. 15, no. 5, May 2004.

[LeSe93] D-S. Lee and B. Sengupta, "Queueing analysis of threshold-based priority scheme for ATM networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, pp. 709-717, 1993.

[LINDO] LINDO Optimization Software, http://www.lindo.com/, June 2004

[Mill60] R.G. Miller, "Priority queues," Ann. Math. Statistics, vol. 31, pp. 86-103.

[Murty95] K. G. Murty, *Operations Research: Deterministic Optimization Models*, Prentice Hall, 1995

[NaBa99] E. Nahum, T. Barzilai, and D. Kandlur, "Performance issues in WWW servers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 1, pp. 216-217, 1999.

[Neut75] M.F. Neuts, "Probability distributions of phase-type," *Liber Amicorum Professor Emeritus H. Florin*, Dept. of Mathematics, University of Louvain, Belgium, pp. 173-206, 1975.

[Neut81] M.F. Neuts, *Matrix Geometric Solutions in Stochastic Models*, Johns Hopkins University Press, Baltimore, 1981.

[NiGe97] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, and C. Lilley, "Network performance effects of HTTP/1.1, CSS1, and PNG," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4, pp. 155-166, 1997.

[PaBa98] R. Pandey, J.F. Barnes, and R. Olsson. "Supporting Quality of service in HTTP Servers," *Proc. the 17th SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pp. 247-256, June 1998.