韓國經營科學會誌
第30권 第2號
2005年 6月

웹 서어버를 위한 유사출생사멸 Threshold 대기행렬모형*

이호우** · 조은성***

A Threshold QBD Queueing Model for Web Server System*

Ho Woo Lee** · Eun-sung Cho***

■ Abstract ■

This paper proposes queueing models for a Web server system which is composed of an infinite-buffer main server and finite-buffer auxiliary server(s). The system is modeled by the level-dependent quasi-birth- death (QBD) process. Utilizing the special structure of the QBD, we convert the infinite level-dependent QBD into a finite level-independent QBD and compute the state probabilities. We then explore the operational characteristics of the proposed web-server models and draw some useful conclusions.

Keyword : Web server system, Queueing model, QBD

1. Introduction

Most of the early web-server systems consisted of a single server which had to process all kinds of user requests and data types. But as the demand for high-quality web services exploded and web technologies advanced, there has been an increasing demand for user-interactive image-based web services.

Usually a user request for a web-page consists of two mixed parts of services : simple HTML-based text part and CGI-based image and applet part (CGI : Common Gate Interface). In the service of the HTML part, the server only needs to

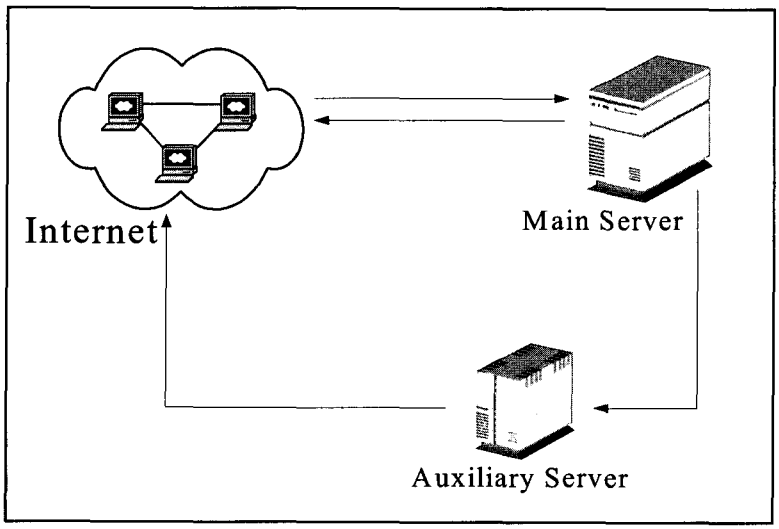
논문접수일 : 2004년 12월 20일 논문게재확정일 : 2005년 3월 25일
* This research was supported by Korea Research Foundation #2003-041-D20555.
** 성균관대학교 시스템경영공학부
*** 성균관대학교 정보통신공학부

send the text pages to the user computer and does not need a large amount of processing time. But, the CGI part needs to retrieve the image and streaming data stored in the hard disk or generate the applets on the user part so that the user can input and retrieve the data to and from the web server. Therefore, a CGI service requires heavy amount of processing time. Moreover, web security is becoming more important in the e-business and the time for the CGI part is getting heavier. As a consequence, with higher probability the single-server web system may not handle the user requests within an appropriate amount of time and on a reasonable level of quality of service (QoS). One simple remedy is obviously to add an auxiliary server. But this requires a careful implementation.

With these observations in mind, we propose the following web-server threshold scheduling policy: the main server (server-1) processes both types of services until the number of requests reaches some pre-determined threshold value N. As soon as the number exceeds N, the processing of the CGI-part of a request is relegated to the auxiliary server (server-2) which has a finite buffer (this buffer size is assumed for control purposes). To prevent excessive overload at server-2, if the buffer is full, the main server does not send any job to the server-2 until there is a space available. Operating in this way, the two servers process the user requests interactively by monitoring the dynamic load levels at each other server [Figure 1.1].

Literature on queueing approach to web-server systems are very rare. Note that we are not dealing with the server switching problem in Internet server clusters (Chase [2]) or load balancing problems in a distributed web-server system (Cardellini, Colajanni and Yu [1]).

Our objectives in this paper are two-fold. Firstly, we want to demonstrate a queueing approach to the analysis of web-server systems. Secondly, we would like to draw some conclusions that might be useful to the cost-effective operation of the web-server systems. We believe that our modeling effort is meaningful



[Figure 1.1] Proposed web-server system

even though the actual user requests for web pages do not follow the Poisson arrival process and the service times are not exactly exponential, because, as widely known in diverse queueing phenomena, overall behavior of queueing processes is generally insensitive to their exact inter-arrival and service time distributions.

In this paper, we first use the infinite level-dependent quasi-birth-death (QBD) process to model the proposed web-server system. We then use the special structure of the QBD to convert it to a finite level-independent QBD and compute the state probabilities. Finally we explore the performance analyses of the proposed web-server systems and draw some meaningful conclusions.

This paper combines the separate works of Lee and Cho [4], brings them under one framework of analysis and provide general interpretations.

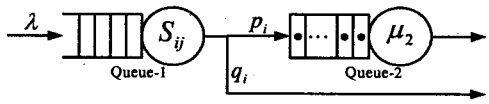
2. Single Auxiliary-Server Model

In this section, we analyze the web server system in which there are a main server and an auxiliary server. We assume that the user requests (customers) arrive according to the Poisson process with rate λ. The normal service rate of server-1 (main server) is exponential with rate μ_1 and that of server-2 (auxiliary server) is μ_2. A new customer is served by server-1 first and then, depending on the service policy, may be transferred to server-2 or leaves the system.

The service policy is as follows. If the queue length (i.e., the number of user requests) is less than or equal to the threshold N, all the customers are served by server-1 at normal rate μ_1 and the finished customers depart the system. If there are more customers than the threshold N at queue-1, the server-1 processes only the HTML-part of a request and if the customer needs an extra processing time for its CGI-part, it is sent to the server-2. In this case, since the processing time of the HTML-part alone is substantially smaller than the service times of the whole part (i.e., HTML+CGI), we increase the service rate of the server-1 from μ_1 to $c\mu_1$, ($c > 1$). We assume that a customer needs both types of services (i.e., HTML and CGI) with probability p, which means that if there are more than N customers at queue-1, each finished customer is sent to queue-2 with probability p. To prevent excessive overload at queue-2, if the buffer is full at queue-2, customers are no longer sent to queue-2 and the service rate of server-1 returns back to the original rate μ_1.

2.1 The Model and Analysis

Let us assume that the buffer capacity is $m-1$ at queue-2 (thus, there can be a maximum of m customers at queue-2 including the one in service). Let us define the system state as (i,j) where i is the queue length at queue-1 and j is the queue length at queue-2. The proposed service policy is depicted in the figure below,

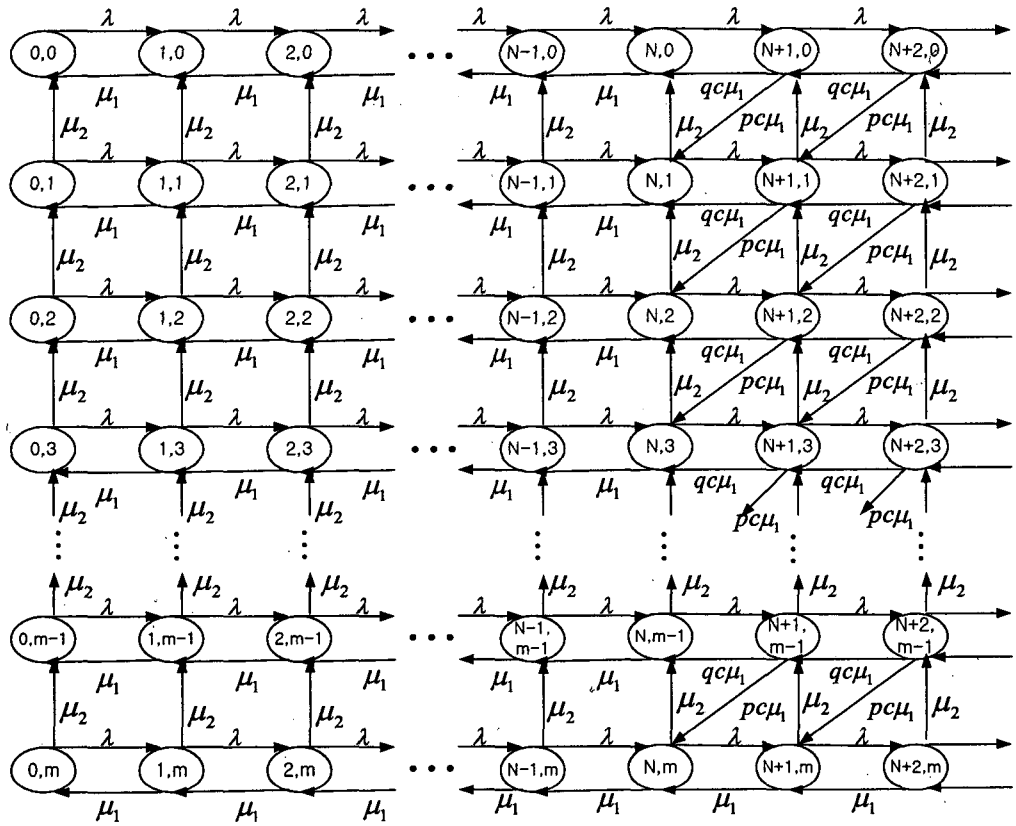


[Figure 2.1] The service process

where the service rates are

$$S_{ij} = \begin{cases} \mu_1, & (i \le N,\ 0 \le j \le m),\ (i > N,\ j = m), \\ c\mu_1, & (i > N,\ 0 \le j \le m-1). \end{cases}$$

The rate-flow diagram of the queue length process is given below.



[Figure 2.2] The transition rate diagram

The infinitesimal generator $\boldsymbol{Q}$ becomes

$$
\mathbf{Q} = \begin{array}{c} \\ l(0) \\ l(1) \\ l(2) \\ \vdots \\ l(N-1) \\ l(N) \\ l(N+1) \\ l(N+2) \\ \vdots \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} l(0) & l(1) & l(2) & l(3) & \cdots & l(N-1) & l(N) & l(N+1) & l(N+2) & l(N+3) & \cdots \end{array} \\
\begin{bmatrix}
\mathbf{B}_0 & \mathbf{A}_0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \hat{\mathbf{A}}_2 & \hat{\mathbf{A}}_1 & \mathbf{A}_0 & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \hat{\mathbf{A}}_2 & \hat{\mathbf{A}}_1 & \mathbf{A}_0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots
\end{bmatrix}
\end{array}
. \tag{2.1}
$$

In the above matrix, $\ell(n)$ denotes level-n. Also, the individual matrices are of size $(m\times m)$ and are as follows :

$$\boldsymbol{B}_0=\begin{pmatrix} -\lambda & 0 & 0 & \cdots & 0 \\ \mu_2 & -(\lambda+\mu_2) & 0 & \cdots & 0 \\ 0 & \mu_2 & -(\lambda+\mu_2) & \cdots & 0 \\ 0 & 0 & \mu_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_2 & -(\lambda+\mu_2) \end{pmatrix}, \tag{2.2}$$

$$\boldsymbol{A}_2=\begin{pmatrix} \mu_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \mu_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \mu_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \mu_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mu_1 \end{pmatrix}, \tag{2.3}$$

$$\boldsymbol{A}_0=\begin{pmatrix} \lambda & 0 & 0 & 0 & \cdots & 0 \\ 0 & \lambda & 0 & 0 & \cdots & 0 \\ 0 & 0 & \lambda & 0 & \cdots & 0 \\ 0 & 0 & 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \lambda \end{pmatrix}, \tag{2.4}$$

$$\boldsymbol{A}_1=\begin{pmatrix} -(\lambda+\mu_1) & 0 & 0 & 0 & \cdots & 0 \\ \mu_2 & -(\lambda+\mu_1+\mu_2) & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mu_2 & -(\lambda+\mu_1+\mu_2) \end{pmatrix}, \tag{2.5}$$

$$\tilde{\boldsymbol{A}}_1=\begin{pmatrix} -(\lambda+c\mu_1) & 0 & 0 & \cdots & 0 \\ \mu_2 & -(\lambda+c\mu_1+\mu_2) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_2 & -(\lambda+\mu_1+\mu_2) \end{pmatrix}, \tag{2.6}$$

and

$$\tilde{\boldsymbol{A}}_2=\begin{pmatrix} qc\mu_1 & pc\mu_1 & 0 & 0 & \cdots & 0 \\ 0 & qc\mu_1 & pc\mu_1 & 0 & \cdots & 0 \\ 0 & 0 & qc\mu_1 & pc\mu_1 & \cdots & 0 \\ 0 & 0 & 0 & qc\mu_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mu_1 \end{pmatrix}. \tag{2.7}$$

Note that starting from level $\ell(N+1)$, the matrices $\boldsymbol{A}_2$ and $\boldsymbol{A}_1$ change to $\tilde{\boldsymbol{A}}_2$ and $\tilde{\boldsymbol{A}}_1$ respectively. This is because the service rate of the server-1 increases from μ_1 to $c\mu_1$ if the queue length at queue-1 exceeds the threshold N. The matrix $\boldsymbol{Q}$ implies that we have an infinite level-dependent QBD but with a very special structure.

For the analysis of the QBD $\boldsymbol{Q}$, we decompose the level space into two groups:

$$\boldsymbol{S}=\{\ell(0),\ell(1),\cdots,\ell(N-1),\ell(N)\},$$
$$\boldsymbol{T}=\{\ell(N+1),\ell(N+2),\ell(N+3),\cdots\}.$$

Then, we have

$$\boldsymbol{Q}=\begin{pmatrix} \boldsymbol{Q}_S & \boldsymbol{Q}_{ST} \\ \boldsymbol{Q}_{TS} & \boldsymbol{Q}_T \end{pmatrix} \tag{2.8}$$

where

$\boldsymbol{Q}_S$ = transition rates between levels within $\boldsymbol{S}$,

$\boldsymbol{Q}_T$ = transition rates between levels within $\boldsymbol{T}$,

$\boldsymbol{Q}_{ST}$ = transition rates from $\boldsymbol{S}$ to $\boldsymbol{T}$, and

$\boldsymbol{Q}_{TS}$ = transition rates from $\boldsymbol{T}$ to $\boldsymbol{S}$.

If we take a closer look at the decomposed infinitesimal generator $\boldsymbol{Q}$, we see that $\boldsymbol{Q}_S$ takes a form of finite level-independent QBD, and $\boldsymbol{Q}_T$, an infinite level-independent QBD. If we utilize these structural characteristics of $\boldsymbol{Q}$, it is possible to convert $\boldsymbol{Q}$ to a finite level-independent QBD and compute the stationary state probabilities $\{\pi_0,\pi_1,\cdots,\pi_N\}$. Then, we can use $\boldsymbol{Q}_T$ to compute $\{\pi_{N+1},\pi_{N+2},\cdots\}$.

For this purpose, we assume that $\boldsymbol{Q}_T$ is positive recurrent. This means

$$\alpha\tilde{\boldsymbol{A}}_2\boldsymbol{e}>\alpha\tilde{\boldsymbol{A}}_0$$

where α is the stationary probability vector of the matrix $\tilde{\boldsymbol{A}}_2+\tilde{\boldsymbol{A}}_1+\boldsymbol{A}_0$. This implies that if the process starts in one of the levels in $\boldsymbol{T}$, it enters $\ell(N)$ in a finite number of transitions. As far as the state probability π_N is concerned, the whole group $\boldsymbol{T}$ can be considered as an

imaginary level $\ell(T)$. Then, we have a new QBD Q_S^* as follows :

$$Q_S^* = \begin{array}{c} \\ \ell(0) \\ \ell(1) \\ \vdots \\ \vdots \\ \ell(N) \\ \ell(T) \end{array} \begin{array}{c} \begin{array}{cccccccc} \ell(0) & \ell(1) & \cdot & \cdot & \cdot & \cdot & \ell(N) & \ell(T) \end{array} \\ \begin{pmatrix} B_0 & A_0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ A_2 & A_1 & A_0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & A_1 & A_0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & A_2 & A_1 & A_0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & A_2 & A_1 & A_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \hat{A}_2 & B_1 \end{pmatrix} \end{array} \quad (2.9)$$

Since the QBD Q is both right and left skip-free, the transitions between S and T occur only through levels $\ell(N)$ and $\ell(N+1)$. Once the process enters $\ell(N+1)$ from $\ell(N)$, it stays in T and returns to $\ell(N)$ through $\ell(N+1)$ again. Thus, the matrix B_1 in (2.9) represents the transitions between the states belonging to the imaginary level $\ell(T)$ which means that B_1 is the infinitesimal generator of the transient continuous-time Markov chain restricted to $\ell(N+1)$ and is given by

$$B_1 = \hat{A}_1 + A_0(-U)^{-1}\hat{A}_2. \quad (2.10)$$

The matrix U in (2.10) is the infinitesimal generator of the transient continuous-time Markov chain that is restricted to level $\ell(N+2)$ before it reaches to $\ell(N+1)$ and is given by (see Latouche and Ramaswami [5])

$$U = \hat{A}_1 + A_0(-U)^{-1}\hat{A}_2. \quad (2.11)$$

Thus we have $$B_1 = U. \quad (2.12)$$

Now, the usual familiar matrices defined on the levels belonging to T can be expressed as

$$R = A_0(-U)^{-1}, \quad (2.13)$$

$$G = (-U)^{-1}\hat{A}_2. \quad (2.14)$$

The (i,j)-element of matrix G is the probability that the process ever enters level $\ell(n-1)$ through phase j under the condition that it starts in phase i of level $\ell(n)$. The (i,j)-element of matrix R is the mean time the process stays in phase j of level $\ell(n+1)$ under the condition that it starts in phase i of level $\ell(n)$. For more detailed definitions and computations of these matrices, readers are referenced to Latouche and Ramaswami [1999].

2.2 Computation of Probabilities

Based on the above analysis, we can compute the state probability vector

$$\pi = \{\pi_0, \pi_1, \cdots, \pi_N, \pi_{N+1}, \cdots\}$$

by taking the steps as follows. Let us denote the prior state probabilities (i.e., before being normalized) by $p = \{p_0, p_1, \cdots, p_N, \cdots\}$. Let us define

$$P_T = \sum_{j=N+1}^{\infty} p_j.$$

(Step 1) Compute U or G and then R.

(Step 2) Starting from p_T, compute p_N and then $\{p_0, p_1, \cdots, p_{N-1}\}$. Since p_T represents the sum of the higher state probabilities from level $\ell(n-1)$, we have the following relationship :

$$p_T = \sum_{j=N+1}^{\infty} p_j = \sum_{i=1}^{\infty} p_N R^i = p_N R(I-R)^{-1} \quad (2.15)$$

which means

$$p_N = p_T \cdot (I-R) \cdot R^{-1}. \quad (2.16)$$

Then, starting from p_N, we can compute $\{p_0, p_1, \cdots, p_{N-1}\}$. Starting vector p_T can be obtained as the level probability vector of $\ell(T)$ in QBD Q_T^*.

(Step 3) Compute the state probabilities $\{\pi_0, \pi_1,$

$\cdots, \pi_N\}$ of the QBD $\boldsymbol{Q}$ by normalizing $\{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_N, \boldsymbol{p}_T\}$,

$$\pi_i = \frac{\boldsymbol{p}_i}{\sum_{i=0}^{N} \boldsymbol{p}_i \mathbf{e} + \boldsymbol{p}_T \mathbf{e}}, \quad (0 \le i \le N) \qquad (2.17)$$

where $\mathbf{e}$ is the column vector of one's.

(Step 4) Compute the state probabilities $\{\pi_{N+1}, \pi_{N+2}, \cdots\}$ of the QBD $\boldsymbol{Q}$ from

$$\pi_j = \pi_N \boldsymbol{R}^{j-N}, \ (j \ge N+1). \qquad (2.18) \blacksquare$$

The theoretical background for (Step 4) is the matrix geometric theory of Neuts [6].

The application of the Linear Level Reduction (LLR) algorithm to the above steps goes as follows. For the LLR algorithm and the matrix C_i used in the following algorithm, readers are referenced to Latouche and Ramaswami [5 : Section 10.1]).

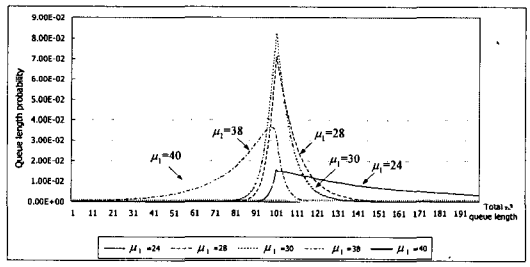
(ALGORITHM)

$\boldsymbol{C}_0 = \boldsymbol{B}_0$;
for $i = 1$ to N, **do**
$\boldsymbol{C}_i = \boldsymbol{A}_1 + \boldsymbol{A}_2(-\boldsymbol{C}_{i-1})^{-1}\boldsymbol{A}_0$;
end
$\boldsymbol{G} = (-\widehat{\boldsymbol{A}}_1)^{-1}\widehat{\boldsymbol{A}}_2$;
while $\| \mathbf{e} - \boldsymbol{G}\,\mathbf{e} \| \ge \varepsilon$, **do**
$\boldsymbol{U} = \widehat{\boldsymbol{A}}_1 + \boldsymbol{A}_0\boldsymbol{G}$;
$\boldsymbol{G} = (-\boldsymbol{U})^{-1}\widehat{\boldsymbol{A}}_2$;
end
$\boldsymbol{B}_1 = \boldsymbol{U}$;
$\boldsymbol{R} = \boldsymbol{A}_0(-\boldsymbol{U})^{-1}$;
$\boldsymbol{C}_T = \boldsymbol{B}_1 + \widehat{\boldsymbol{A}}_2(-\boldsymbol{C}_N)^{-1}\boldsymbol{A}_0$;
solve
$\boldsymbol{p}_T \cdot \boldsymbol{C}_T = 0, \ \boldsymbol{p}_T \cdot \mathbf{e} = 1$;
end
$\boldsymbol{p}_N = \boldsymbol{p}_T \cdot (\boldsymbol{I} - \boldsymbol{R}) \cdot \boldsymbol{R}^{-1}$;
for $i = N-1$ back to 0, **do**
$\boldsymbol{p}_i = \boldsymbol{p}_{i+1}\boldsymbol{A}_2(-\boldsymbol{C}_i)^{-1}$;
end
$\pi_i = (\boldsymbol{p} \cdot \mathbf{e})^{-1}\boldsymbol{p}_i, \ (0 \le i \le N)$
$\pi_j = \pi_N \boldsymbol{R}^{j-N}, \ (N+1 \le j)$

2.3 Performance Analysis

Based on the above analysis, we evaluate the system performance of the proposed web-server model. Assume that the maximum queue length at queue-2 is 200 ($m = 200$) and the threshold at server-1 is $N = 100$. Also, we assume $\lambda = 40$, $\mu_2 = 20$, $c = 2$, and $p = 0.6$.

[Figure 2.3] shows the queue length probabilities of the total number of customers for different values of μ_1.

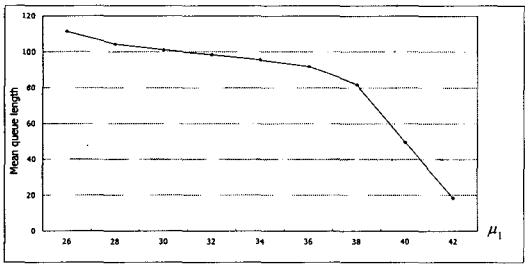


[Figure 2.3] Queue length probability for different μ_1 ($m = 200, N = 100, \lambda = 40, \mu_2 = 20, c = 2, p = 0.6$)

For all cases, the maximum queue length probability is reached at around 100 customers. This is because until the queue length reaches the threshold $N = 100$, the service rate at server-1 is relatively low and customers begin to be dispatched to queue-2 only when the queue length exceeds the threshold. It is interesting to see the waves rising from the right and waning to the left as μ_1 increases. The height of the wave reaches the peak when the queue length reaches 100. On the other hand and as can be expected, if μ_1 is very high (see the graph of $\mu_1 = 40$), the

queue length probabilities begin to fall at around the threshold $N=100$. This is because just after the queue length reaches the threshold, the service rate at server-1 increases to $c\mu_1$ and the server-2 joins to operate.

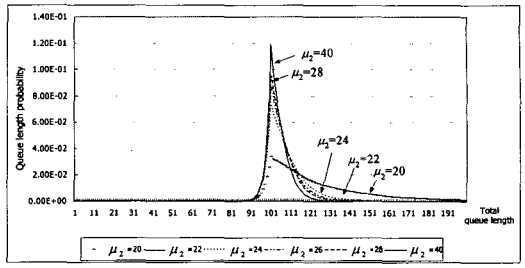
[Figure 2.4] shows the mean queue length when μ_1 varies. It can be seen that the mean queue length drops abruptly when μ_1 is around 38 under the current parameter setting. Note that this is the service rate at around the end of the waves in [Figure 2.3].



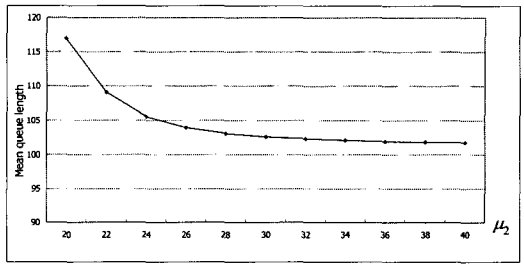
[Figure 2.4] Mean queue lengths as μ_1 varies
($m=200, N=100, \lambda=40, \mu_2=20, c=2, p=0.6$)

[Figure 2.5] shows the change of the queue length probabilities for different values of μ_2 when μ_1 is fixed. Note that the queue length probabilities drop quickly for all cases at around 100, which is different from [Figure 2.3]. The slope of the mean queue length changes slowly ([Figure 2.6]). This is due to the fact the server-2 has a finite buffer size and the customers are sent to the server-2 only when the queue length at server-1 exceeds the threshold.

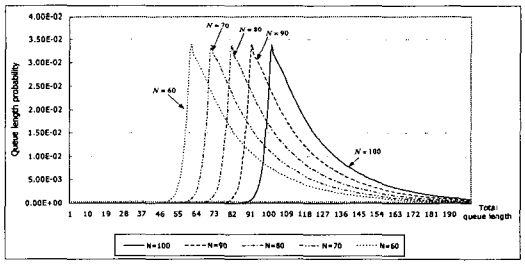
[Figure 2.7] shows the queue length probabilities as the threshold changes. For all cases, the maximum queue length probability is reached at around each threshold. [Figure 2.8] shows the mean queue length as the threshold varies. It is interesting to see that the mean queue length lthe queue length probabilities at [Figure 2.7] show



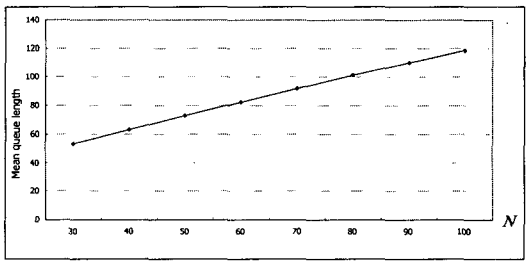
[Figure 2.5] Queue length probability for different μ_2
($m=200, N=100, \lambda=40, \mu_1=25, c=2, p=0.6$)



[Figure 2.6] Mean queue lengths as μ_2 varies
($m=200, N=100, \lambda=40, \mu_2=20, c=2, p=0.6$)



[Figure 2.7] Queue length probabilities when N varies
($m=200, \lambda=40, \mu_1=25, \mu_2=20, c=2, p=0.6$)



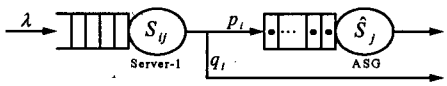
[Figure 2.8] Mean queue length when N varies
($m=200, \lambda=40, \mu_1=25, \mu_2=20, c=2, p=0.6$)

the identical shapes and behavior (but with dif-

ferent mean values).

3. Multiple Auxiliary-Server Model

This section extends the work of section 2 to the case of multiple auxiliary servers in which there are a main server and an auxiliary server group (ASG). The ASG is assumed to consist of k identical servers. The system operation is depicted in the figure below,



[Figure 3.1]

where the service rates are given by

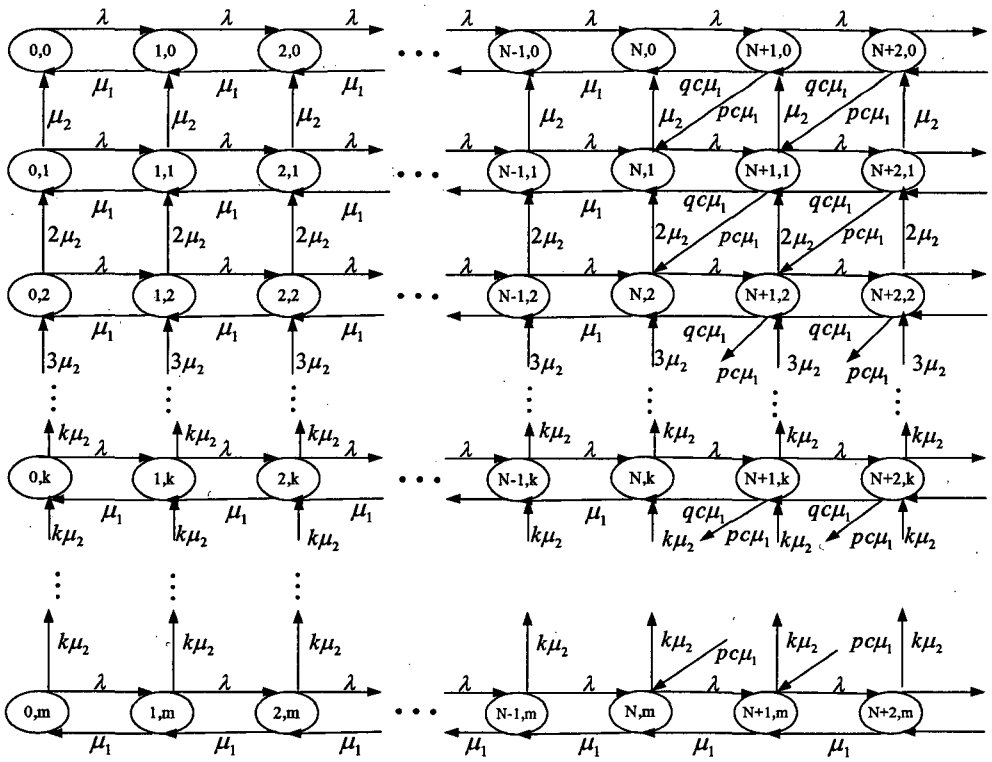
$$S_{ij}=\begin{cases}\mu_1, & (i\le N,\ 0\le j\le m),\ (i>N,\ j=m),\\ c\mu_1, & (i>N,\ 0\le j\le m-1),\end{cases}$$

$$\Im_j=\begin{cases}j\mu_2, & (1\le j\le k-1),\\ k\mu_2, & (k\le j\le m).\end{cases}$$

The service rate $\Im_j$ came from the fact that the $M/M/k$ queueing system can be modeled as an $M/M/1$ with state-dependent service rate. In the figure, p_i and q_i are given by

$$p_i, q_i=\begin{cases}p_i=0,\ q_i=1, & (i\le N),\\ 0<p_i\le 1,\ q_i=1-p_i, & (i>N).\end{cases}$$

The transition rate diagram is as follows :



[Figure 3.2] Transition rate diagram

The infinitesimal generator Q is of the same form with (2.1) where,

$$B_0=\begin{pmatrix} -\lambda & 0 & 0 & 0 & \cdots & 0 & 0 \\ \mu_2 & -(\lambda+\mu_2) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2\mu_2 & -(\lambda+2\mu_2) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & 0 \\ 0 & 0 & 0 & k\mu_2 & -(\lambda+k\mu_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & k\mu_2 & -(\lambda+k\mu_2) \end{pmatrix}, \tag{3.1}$$

$$A_2=\begin{pmatrix} \mu_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \mu_1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \mu_1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \mu_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mu_1 \end{pmatrix}, \tag{3.2}$$

$$A_0=\begin{pmatrix} \lambda & 0 & 0 & 0 & \cdots & 0 \\ 0 & \lambda & 0 & 0 & \cdots & 0 \\ 0 & 0 & \lambda & 0 & \cdots & 0 \\ 0 & 0 & 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \lambda \end{pmatrix}, \tag{3.3}$$

$$A_1=\begin{pmatrix} -(\lambda+\mu_1) & 0 & 0 & 0 & \cdots & 0 & 0 \\ \mu_2 & -(\lambda+\mu_1+\mu_2) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2\mu_2 & -(\lambda+\mu_1+2\mu_2) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & k\mu_2 & -(\lambda+\mu_1+k\mu_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & k\mu_2 & -(\lambda+\mu_1+k\mu_2) \end{pmatrix}, \tag{3.4}$$

$$\tilde{A}_1=\begin{pmatrix} -(\lambda+c\mu_1) & 0 & 0 & 0 & \cdots & 0 & 0 \\ \mu_2 & -(\lambda+c\mu_1+\mu_2) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2\mu_2 & -(\lambda+c\mu_1+2\mu_2) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & k\mu_2 & -(\lambda+c\mu_1+k\mu_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & k\mu_2 & -(\lambda+\mu_1+k\mu_2) \end{pmatrix}, \tag{3.5}$$

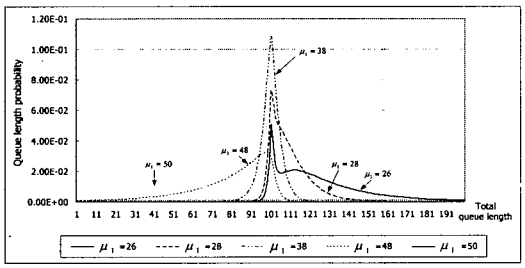
and

$$\tilde{A}_2=\begin{pmatrix} qc\mu_1 & pc\mu_1 & 0 & 0 & \cdots & 0 \\ 0 & qc\mu_1 & pc\mu_1 & 0 & \cdots & 0 \\ 0 & 0 & qc\mu_1 & pc\mu_1 & \cdots & 0 \\ 0 & 0 & 0 & qc\mu_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mu_1 \end{pmatrix}. \tag{3.6}$$

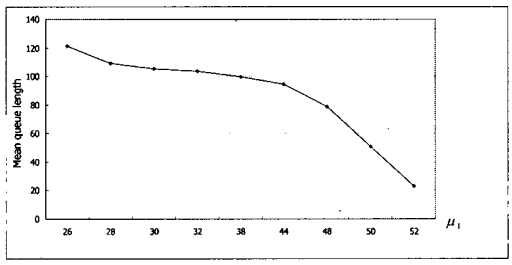
The computation procedure is exactly the same as in the single auxiliary-server model of section 2.2.

3.1 Performance analysis

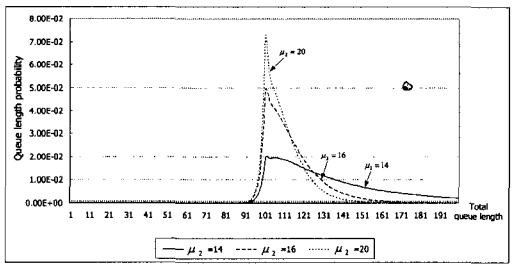
[Figure 3.3]~[Figure 3.8] show the performance of the system with two auxiliary servers. As can be seen in the figures, the overall behavior is very similar to that of the single auxiliary-server system.



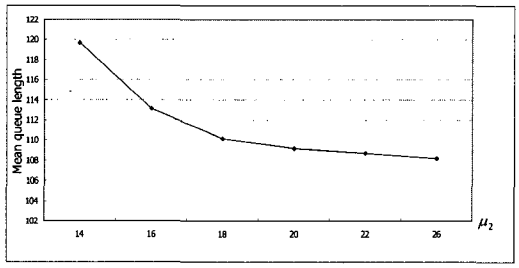
[Figure 3.3] Queue length probability for different μ_1 ($m=200, N=100, \lambda=50, \mu_2=20, c=2, p=0.6, k=2$)



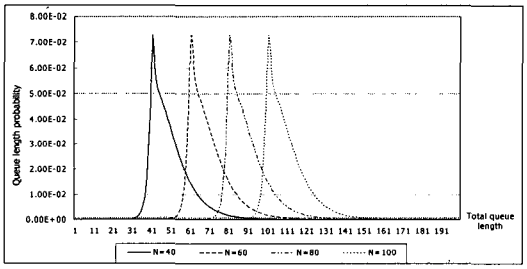
[Figure 3.4] Mean queue length for different μ_1 ($m=200, N=100, \lambda=50, \mu_2=20, c=2, p=0.6, k=2$)



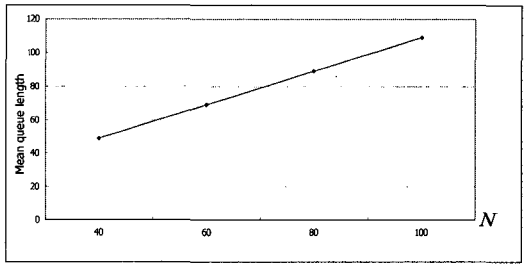
[Figure 3.5] Queue length probability for different μ_2 ($m=200$, $N=100$, $\lambda=50$, $\mu_1=28$, $c=2$, $p=0.6$, $k=2$)



[Figure 3.6] Mean queue length for different μ_2 ($m=200$, $N=100$, $\lambda=50$, $\mu_1=28$, $c=2$, $p=0.6$, $k=2$)



[Figure 3.7] Queue length probabilities when N varies($m=200$, $\lambda=50$, $u_1=28$, $\mu_2=20$, $c=2$, $p=0.6$, $k=2$)



[Figure 3.8] Mean queue length when N varies ($m=200$, $\lambda=50$, $u_1=28$, $\mu_2=20$, $c=2$, $p=0.6$, $k=2$)

4. Conclusions Drawn from the Performance Analyses

From the analyses and interpretations of the operational characteristics so far, we can draw the following conclusions :

(1) The effect of the service rate at server-2 is less significant than the service rate at server-1 on the system performance. This implies that if one wants to control the system behavior within a limited budget, it would be profitable to exert his effort to controlling server-1 rather than server-2. Along the same line, if both the threshold N at server-1 and the buffer size m at server-2 are controllable, then, controlling m may not be so fruitful than controlling the threshold N at server-1.

(2) The wave pattern of the queue length probabilities when μ_1 changes may serve as an important indicator of more crucial features concerning the system performance (see [Figures 2.3], [Figure 2.5], and [Figures 3.3], [Figures 3.5]). Thus, it may be meaningful to keep track of the change of the queue length probability patterns at queue-1.

(3) Above (1) and (2) give us an important tip concerning the cost-effective operation of the web-server system : adding a more powerful server-2 than the main server may not be fruitful in the operation of the whole web-server system. If the server-2 is more powerful, then it should be diverted to the main server.

5. Comments and Summary

In this paper, we proposed a queueing model for a web-server system. We used the quasi-birth-death (QBD) queueing model to develop an algorithm and compute the queue length probabilities. Under various parameter settings, we evaluated the operational characteristics of the web-server system. Our study shows that the service rate at server-1 is more significant in the behavior of the whole system. Thus, if any effort needs to be made to control the system performance, it should be directed to the server-1.

If the number of servers at queue-1 is multiple, we would have level-dependent QBD even before the level N. But we can still use the same algorithm of section 2 but with slightly modified matrices $C^{(i)}$ for each level i. The performance analysis of this system will be left as a future research.

References

[1] Cardellini, V., M. Colajanni and P.S. Yu, "Dynamic load balancing on Web-server systems," *IEEE Internet Computing*, Vol.3, Issue 3(1999), pp.28-39.

[2] Chase, J.S., *Server switching : yesterday and tomorrow, Internet Applications, WIAPP Proceedings*, The Second IEEE Workshop, 2001, pp.114-123.

[3] Lee, H.W. and E.S. Cho, A queueing model for web-server system, Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, July 2004a, pp.379-383.

[4] Lee, H.W. and E.S. Cho, A multi-server queueing model for web-server system, Proceedings of the 34th International Conference on Computers and Industrial Engineering, San Francisco, Nov. 2004b, pp.472-477.

[5] Latouche, G. and V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, (ASA-SIAM series on statistics and applied probability), 1999.

[6] Neuts, M.F., Matrix-geometric solutions in Stochastic Models, The Johns Hopkins University Press., Baltimore, MD, 1981.