Performance Analysis of a Web Server Supporting Multiple Classes of Requests

Sai Rajesh Mahabhashyam and Natarajan Gautam
310 Leonhard Building, The Pennsylvania State University, University Park, PA, 16802, USA

Abstract

We consider a web server that supports two classes of requests, streaming and elastic. Due to the varying Quality of Service requirements and analytical complexity involved, there is little research that considers a single model capturing the needs of all classes of requests. The shortcoming in addressed as follows: (i) a matrix geometric method is used to analytically derive performance measures, (ii) a mathematical model is formulated that caters to the multiple objectives, (iii) a cost model is developed to obtain an admission control policy that utilizes the processing capacity optimally. Further, we compare our policy against existing priority and resource partitioning policies.

Keywords: Stochastic Processes / Queuing Theory, Performance Evaluation, Resource Sharing.

1 Introduction

On one hand the Internet users and applications are growing at an exponential rate and on the other hand the economy is slowing down tremendously. This has created a shift in paradigm in the way organizations view resources. In particular, companies are focusing on utilizing resources efficiently and squeezing the most out of them. This is done not only spatially where bandwidth, buffer space, etc. are effectively consumed, but also temporally by appropriately tuning resources over time under varying load conditions. In fact, with the emphasis on energy conservation, it is now critical to not only plan for resources during peak periods but also optimally turn resources off during lean periods. We address a piece of this important problem by considering a webserver that supports multi-class requests and developing techniques to utilize processing capacity effectively over time.

We consider multiple classes of requests with varying Quality of Service (QoS) needs. Specifically, there are two main request classes, one is a streaming class that has bandwidth requirements, and the other is elastic class that utilizes the processing capacity not used by the streaming requests. The crucial point is that the objectives for both classes are considered for optimization, i.e. the blocking probability of streaming requests and the delay of elastic requests are both minimized. The motivation for this comes from the fact that most web pages have both streaming as well as elastic requests that need to be delivered to the users. Therefore it is critical to give importance to both classes of requests. In networking community, the literature especially on analytical models for multi-class requests, falls under one of two categories: loss networks and delay networks.

Researchers focusing on *loss networks* consider only the objectives of the streaming traffic and ignore the needs of the elastic traffic. Several articles (such as [11], [10] and [2]) on the stochastic knapsack problem, $G/G/C/C$ queueing models, etc., deal with the streaming traffic alone and solve several optimization problems with the understanding that any unused bandwidth will be consumed by elastic traffic. However, these articles do not consider the needs of the data traffic. Further, researchers that focus on *delay networks* assume that a portion of bandwidth is reserved for streaming traffic and the remaining is consumed by the elastic traffic. Thereby these researchers focus on obtaining the delay faced by elastic traffic and ignore any available resources not used by the streaming traffic.

One of the caveats of combining loss and delay networks is to devise a common denominator for the analysis. In that light, we use bandwidth to analyze data traffic following [5] where they model data file sizes (as opposed to transmission time) according to an exponential distribution. If the transmission speed is not constant (due to varying bandwidth availability), then the time to transmit the data is not necessarily exponential. Another caveat as pointed by [1] is that even with two distinct streaming classes with priorities it is not possible to obtain closed-form algebraic expressions for performance measures. Similarly, we modeled and found that it is not possible to obtain closed-form expressions for the performance measures.

The rest of the paper is organized as follows. In section 2, the problem that we are considering is explained with all the notation used. Also, we define the measures that we will be using to evaluate the performance of the system. In

Table 1: Notation

Var	Explanation
λ_1	Arrival rate of class 1 requests
λ_2	Arrival rate of class 2 requests
$1/\mu_1$	Average holding time of class 1 request
$1/\mu_2$	Average class 2 file size
n	Maximum no. of class 1 requests allowed simultaneously
b	Bandwidth allocated to each class 1 request when admitted
c	Minimum bandwidth for class 2 requests
S	Total processing capacity of the web server
$X(t)$	No. of class 1 requests ongoing at time t
$Y(t)$	No. of class 2 requests ongoing at time t
θ_i	Rate of completion of class 2 requests when there are i requests
p_n	Steady state probability that there are n class 1 ongoing requests
W	Average delay of class 2 requests
$S^{(i)}$	State space with i class 2 requests
$\pi^{(i)}$	Steady state probability vector for state space $S^{(i)}$
R	Auxilliary matrix in matrix geometric method

Section 3, we develop an analytical model of the process and provide the formula to calculate the loss probability of streaming requests. In section 4, we discuss a method to calculate the average delay of the elastic requests. In section 5, a mathematical model with multiple objectives is formulated and a cost model is developed to combine multiple objectives into a single objective to obtain an admission control policy that utilizes the processing capacity in an optimal manner. Single-period analysis is also discussed with examples. Finally, in section 6, we present concluding remarks and possible extensions of this work.

2 Problem Definition

2.1 Notation

Consider a web server with the following "static" admission control policy. The maximum number of class 1 requests allowed simultaneously is 'n'. That is, an arriving class 1 request is rejected if there are 'n' ongoing class 1 requests at that time, else the arriving request is admitted and allocated a fixed bandwidth 'b'. All arriving class 2 requests are admitted into the system. That is accomplished by choosing $c > 0$ such that $S = nb + c$, where S is the processing capacity of the webserver.

At any given time t, let $X(t)$ and $Y(t)$ be number of ongoing class 1 and class 2 requests respectively. We consider the following processor sharing mechanism for class 2 requests. Class 2 requests share the processing capacity not used by class 1. Therefore, since the unused capacity after allocating $X(t)$ class 1 requests is $R - b\,X(t) = [n - X(t)]b + c$, each class 2 request is allocated a bandwidth $[[n - X(t)]b + c]/Y(t)$, whenever $Y(t) > 0$. The notation used above and throughout the paper is summarized in Table 1.

2.2 Performance measures

1. Loss probability of class 1 requests: A request is rejected or a call is blocked if and only if there are 'n' ongoing calls at the time of arrival of that new call. The probability of a call getting rejected is called *loss probability* or *blocking probability*. In steady state, under certain conditions such as homogenous arrivals, it is equal to the fraction of time when the system has 'n' class 1 calls.

2. Average delay of class 2 requests: There is no loss or blocking in class 2 as all the calls are accepted but the time to transfer the file increases when new calls arrive. Hence, the measure chosen to represent the performance of class 2 is '*average latency*' or '*average delay*'. The service rate changes dynamically and is dependent on the number of the class 1 users at that instant. So it is a non-trivial task to calculate average delay a class 2 user experiences.

3 Analytical Model

Let the request arrivals in both class 1 and class 2 be according to Poisson processes with rates λ_1 and λ_2 respectively. Also, let the holding time of class 1 requests be exponentially distributed with mean $1/\mu_1$ and let the file size of class 2 be exponentially distributed with mean size $1/\mu_2$. The exponential distribution is assumed so as to keep the analysis tractable.

3.1 Blocking probability of class 1

To obtain the blocking probability of class 1 requests, we only need to consider transitions of class 1 requests as they are not affected by the class 2 requests. Note (from Table 1) that $X(t)$ is the number of ongoing class 1 requests at time t. The process $\{X(t), t \geqslant 0\}$ is modelled as a Continuous Time Markov chain (CTMC) with transitions out of state $X(t) = i$ represented as $i \rightarrow i+1: \ \lambda_1$ and $i \rightarrow i-1: \ i\mu_1$. This is a birth and death process with birth rate λ_1 and death rate $i\mu_1$. From a queuing standpoint, this is a standard $M/M/n/n$ system and the loss probability is given by $p_n = \frac{1/n!(\lambda_1/\mu_1)^n}{\sum_{k=0}^{n} 1/k!(\lambda_1/\mu_1)^k}$ where p_n denotes the steady state probability that there are n ongoing class 1 requests. This is the well known Erlang B formula ([11], [3]) for the $M/M/n/n$ queuing system. In our analysis, we consider holding times to be exponentially distributed in order to keep the class 2 analysis tractable. We now shift our attention to the other performance measure stated in section 2, viz., average latency or delay of class 2 calls.

3.2 Bivariate stochastic process

For class 2 analysis, it is important to consider class 1 as well due to the inherent dependence. As stated in 2.1, let $X(t)$ and $Y(t)$ be the number of class 1 and class 2 requests respectively ongoing at time t. The bivariate stochastic process $\{(X(t), Y(t)), t \geq 0\}$ is a CTMC. The value of $X(t)$ varies from 0 to n where as $Y(t)$ can be anywhere from 0 to ∞. So the state space S = {(0,0),(1,0),....(n,0), (0,1), (1,1),....(n,1), (0,2),(1,2),....(n,2).....}. For $0 < i < n$ and $j > 0$, the transition rates are as follows: $(i,j) \rightarrow (i+1,j): \lambda_1$, $(i,j) \rightarrow (i-1,j): i\mu_1$, $(i,j) \rightarrow (i,j+1): \lambda_2$ and $(i,j) \rightarrow (i,j-1): \mu_2[(n-i)b+c]$. The first three transitions are relatively straight forward. The transition rate from (i,j) to $(i,j-1)$ can be obtained as follows. The available bandwidth for each of j files in class 2 when there are i class 1 requests is $[(n-i)b+c]/j$. So the average rate of transfer of a single class 2 file is $\mu_2[(n-i)b+c]/j$. So the transition rate from j to $j-1$ is $j[\mu_2[(n-i)b+c]/j]$, which is $\mu_2[(n-i)b+c]$ and is denoted by θ_i. It is easy to show that θ_i is also exponentially distributed. When $i = 0$ or $i = n$, there are no transitions from i to $i-1$ or i to $i+1$ respectively. Also, when $j = 0$, there is no transition from j to $j-1$ and The transition rates are adjusted accordingly. Thus, the above system has been modelled as 2-dimensional infinite state CTMC.

4 Calculation of Average Delay for Class 2

There does not exist any closed form solution for the steady state joint probability $\{X(t), Y(t)\}$. Therefore, the average delay of class 2 requests cannot be obtained in closed form. The bivariate stochastic process modelled in Section 3.2 is a Quasi-birth-death (QBD) process. Since QBD has a special structure, a technique called Matrix Geometric Method (MGM) can be used to solve that 2-dimensional CTMC. The general theory of MGM and how it is applied to our problem is explained in Section 4.2

4.1 Quasi-Birth and death process

As per [4], the definition for quasi birth-death process is as follows*:*

Definition: A continuous time Quasi-Birth-Death (QBD) process is a continuous time Markov process whose infinitesimal generator matrix is of the block partitioned form

$$\mathbf{Q_{QBD}} = \begin{pmatrix} B_1 & A_0 & 0 & 0 & 0 & \dots \\ A_2 & A_1 & A_0 & 0 & 0 & \dots \\ 0 & A_2 & A_1 & A_0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

where A_2, A_1 and A_0 and B_1 are matrices.

After partitioning the states into subsets $l(i) = \{(i,j); i \geq 0, 1 \leq j \leq m\}$ called levels, position j within the level is termed as phase. The process can jump down one level, stay in the same level or jump up one level and the rate that these transitions occur are given by A_2, A_1 and A_0 respectively. The process is said to be skip free between levels.

4.2 Matrix Geometric Method for QBD

Matrix geometric methods are widely used for the exact analysis of a general and frequently encountered type of queuing models. The MGM can only be applied if the system can be decomposed into two parts: the initial portion and the repetitive portion. One main advantage is that it can overcome the rapid growth of the state space while constructing the infinitesimal generator matrix of the birth-death process.

There exists a matrix geometric relation among the stationary probabilities which simplifies the algebraic expressions. In matrix geometric method, an auxiliary matrix called R is used in the calculation of stationary probabilities and other measures of interest like waiting time and mean queue length. We have a level-independent infinite-level quasi-birth-death process. The main step here is to find a matrix called R which is used in the computation of the steady state probability vector and other measures of interest.

Computation of matrix R: The matrix R have a following quadratic relation ([8], [9]): $A_0 + RA_1 + R^2A_2 = 0$. Neuts [7] defines infinite-state Markov chains with a repetitive structure with state space partitioned into the boundary states $S^{(0)} = \{s_0^0, ..., s_0^n\}$ and a set of states $S^{(i)} = \{s_i^0, ..., s_i^n\}$, that correspond to the repetitive portion of the chain. Let $\pi^{(i)}$ be the steady state probability vector of states $S^{(i)}$. Then $\pi^{(i)} = \pi^{(1)} R^{i-1} \ \forall \ i \geq 1$. Solving $\pi Q_{QBD} = 0$ will give both $\pi^{(0)}$ and $\pi^{(1)}$. The following set of equations are obtained.

$$\begin{aligned} \pi^{(0)}B_0 + \pi^{(1)}A_2 &= 0 \\ \pi^{(0)}A_0 + \pi^{(1)}(A_1 + RA_2) &= 0 \\ \pi^{(0)}e + \pi^{(1)}(I-R)^{-1}e &= 1 \end{aligned} \tag{1}$$

Once $\pi^{(0)}$ and $\pi^{(1)}$ are obtained, the expected holding time of a job in the system can be calculated as follows: $W = \lambda_2^{-1}(\pi^1(I-R)^{-1}e + \pi^1 R(I-R)^{-2}e)$ where W is the average delay or holding time and e is the column vector of ones.

We apply the above method to our problem. Recall A_0, A_1, A_2 and B_1, which were defined as the elements of Q_{QBD} earlier in this section. The structure of the infinitesimal generator for this QBD problem is shown below:

$$\mathbf{A_0} = \begin{pmatrix} \lambda_2 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ 0 & 0 & \dots & 0 & \lambda_2 \end{pmatrix}, \mathbf{A_1} = \begin{pmatrix} -\lambda_1 - \theta_0 & \lambda_1 & 0 & \dots & 0 \\ \mu_1 & s(1) & \lambda_1 & \dots & 0 \\ 0 & 2\mu_1 & s(2) & \lambda_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & n\mu_1 & s(n) \end{pmatrix},$$

$$\mathbf{A_2} = \begin{pmatrix} \theta_0 & 0 & 0 & \dots & 0 \\ 0 & \theta_1 & 0 & \dots & 0 \\ 0 & 0 & \theta_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \theta_n \end{pmatrix} \text{ and } \mathbf{B_1} = \begin{pmatrix} -\lambda_1 & \lambda_1 & 0 & \dots & 0 \\ \mu_1 & u(1) & \lambda_1 & \dots & 0 \\ 0 & 2\mu_1 & u(2) & \lambda_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & n\mu_1 & u(n) \end{pmatrix}$$

where $s(i) = -\lambda_1 - \theta_i - i\mu_1$and $u(i) = -\lambda_1 - i\mu_1$. Thus, substituting the above matrices in the set of equations (1), we get $\pi^{(0)}$ and $\pi^{(1)}$. Then, we obtain the average delay for class 2 requests .Thus, substituting the above matrices in the set of equations (1), we get $\pi^{(0)}$ and $\pi^{(1)}$. Then, we obtain the average delay for class 2 calls.

5 Resource Optimization

5.1 Choosing optimal n

The analysis so far has been done assuming that all parameters are given and are fixed. It is reasonable to consider S and b to be fixed by infrastructure and applications respectively. However, n could be chosen such that the processing capacity is utilized optimally. Note that once n is determined $c = S - nb$. If n is increased, there is a decrease in c and thus the probability that class 1 user is rejected decreases while average class 2 file transfer time increases.

There are two objectives viz., minimize the loss probability of class 1 requests and minimize the average delay of class 2 requests. Thus, it becomes a multi-criteria decision model with two conflicting objectives. With increase in

n, the loss probability of class 1 requests reduces but at the same time the average delay of class 2 requests increases. The two objectives can be combined into one objective of maximizing the total revenue or minimizing the total cost incurred by rejecting a class 1 request or delaying the class 2 request for one more unit time. We attach some weights that translate to price or penalty for a rejection of class 1 request and unit delay cost for class 2 request. Let r be the cost of rejection of a class 1 request and let h be the holding cost per unit time of delay of class 2 request. Then, the total cost incurred per unit time is given by $C = \lambda_1 \, r \, p_n \, + \, \lambda_2 \, h \, W$. It can be observed that at some $n = n^*$, the cost C is minimized. In this way, we can choose optimal n value to minimize the total cost.

5.2 Single Period Analysis

We compare our policy with two other policies. The first policy (Priority policy, Section 5.2.1), is based on the loss networks where only the bandwidth-sensitive class 1 requests is considered. The second policy (Resource partitioning policy, section 5.2.2) is based on research in the area of delay networks where the bandwidth S is partitioned for the two classes. It is to be noted that the priority policy and resource partitioning policy are a consequence of the current modelling approaches, however they have not been explicitly considered in the literature. The main purpose is to illustrate the benefits of using our policy.

In our policy, class 2 utilizes the unused bandwidth left by class 1. The purpose of this policy is to utilize the bandwidth efficiently and also provide QoS to both the classes simultaneously. The processing capacity is taken as $S = 24$ and the bandwidth is taken as $b = 5$ in all the cases. The costs r and h are taken to be 5 and 1 respectively. We have looked at many examples and all of them indicate that our policy performs better than the other two policies. But due to lack of space, we present only a few examples to illustrate that.

5.2.1 Policy 1: Priority

In this policy, class 1 is given absolute priority over class 2. The class 2 requests are ignored and only class 1 requests are considered important with respect to QoS. The total bandwidth is allocated such that all class 1 requests are accepted until no more can be accepted. That is, n is the largest integer such that $n.b < S$ or $n = \lfloor \frac{S}{b} \rfloor$ denoted by n_1^*. The remaining bandwidth is allocated to class 2 and no QoS is provided to class 2 users.

Table 2 illustrates how our policy does better than the policy in which absolute priority is given to class 1. Let n^* and $mincost$ denote the optimum n value and cost at that optimum n respectively. Similarly $mincost1$ denotes the cost when $n = n_1^*$ for priority policy. As the average file size increases, it can be seen that $mincost$ increases at a steady rate whereas $mincost1$ increases rapidly. It can also be seen that $mincost$ is always less than $mincost1$ in all cases.

5.2.2 Policy 2: Resource partitioning

In this policy, class 1 and class 2 are dealt separately. The total bandwidth is split into two partitions: class 1 is allocated a bandwidth of $n.b$ and class 2, a bandwidth of c. The bandwidth unused by class 1 is not utilized by class 2. Thus, class 2 will always have a bandwidth of exactly c.

Table 3 shows how our policy does better than this case. Here, n_2^* is the optimal value of n where we get minimum cost (denoted by $mincost2$) with the partitioned policy. In most of the cases, it can be seen that $mincost2$ is far more than $mincost$. That means, our policy performs a lot better than resource partitioning policy also.

6 Concluding remarks

In this paper, we considered a web server with two classes viz., bandwidth-sensitive or streaming requests and elastic requests that utilizes the unused processing speed by the real-time requests. The performance measures chosen were loss probability for elastic requests and average delay for the data traffic. The paper focusses on obtaining the average delay of the data traffic. Matrix geometric method was used to analytically solve a 2-dimensional infinite state Markov chain. A multi-objective optimization model involving the performance measures was developed and converted into a single objective cost model.

The following are some possible extensions. It is quite common that the arrival rates change according to time of the day and day of the week ([6]). We have peak and non-peak periods when the parameters like arrival rate change

Table 2: Policy 1: Absolute priority for class 1

λ_1,μ_1, λ_2, μ_2	n^*	mincost	n_1^*	mincost1
3, 1, 20, 2.5	3	7.06	4	9.13
3, 1, 20, 2.0	3	9.05	4	19.94
3, 1, 20, 1.9	2	9.81	4	27.05
3, 1, 20, 1.8	2	10.14	4	41.33
3, 1, 20, 1.7	2	10.59	4	76.23
3, 1, 20, 1.6	2	11.27	4	185.84

Table 3: Policy 2: Partitioned processing speed

λ_1,μ_1, λ_2, μ_2	n^*	mincost	n_2^*	mincost2
2, 1, 10, 2	4	1.80	2	9.56
2, 1, 20, 2	3	4.74	1	28.89
2, 1, 10, 0.8	2	6.61	1	25.89
2, 1, 30, 2	1	9.50	1	119.17
6, 1, 3, 2	4	14.39	4	15.89
6, 1, 10, 2	4	16.97	2	27.16

over time. Then, based on the above optimization procedure, the value of n can be optimized after every time interval autonomically. The data of the arrival times for a small window of time can be taken and updated arrival rates can be calculated. The time interval should not be very big so that we have stale information. At the same time, we assume that the time period is long enough that steady state is reached.

It has been assumed till now that all class 2 requests are admitted at all times. A variation of the problem is to have some admission control policy for admitting class 2 requests. The acceptance or rejection of requests of both classes can be interactively controlled. Based on the number of class 1 and 2 requests and average delay experienced by class 2 requests, the admission of arriving requests of both the classes can be controlled. In the problem, the file size distribution is assumed to be exponential. So, similar analysis could be tried if the file size is a non-exponential. It may be possible to extend MGM using phase type distributions as well.

Acknowledgments

The authors thank Prof. G. Kesidis, Prof. R. Acharya, Dr. D. Ghosh and Prof. V. Sarangan for initial discussions that led to the formulation of this research. This research was partially supported by NSF Grant No. ANI-0219747.

References

[1] Greenberg, A. G., Srikant, R., and Whitt, W, 1999, "Resource sharing for book-ahead and instantaneous-request calls", IEEE/ACM Transaction on Networking, vol. 7, no.1, pp. 10-22, Feb 1999.

[2] Kaufman, J. S., 1981, "Blocking in Shared Resource Environment", IEEE Transactions on Communications, vol. 29, no. 10, pp. 1494-1481.

[3] Kulkarni, 1995, V. G., Modeling and Analysis of Stochastic Systems. Chapman and Hall texts in Statistical series, CRC press.

[4] Latouche, G., Pierce, C., and Taylor, P., 1997, "Invariant measures for quasi-birth and death processes", Stochastic Models.

[5] Mandjes, G., Mitra, D., and Scheinhardt, W., 2002, "Simple models of network access, with application to the joint rate and admission control", Proc. INFOCOM, vol. 1, pp. 3-12.

[6] Massey, W., 2002, "The analysis of queues with time-varying rates for telecommunication models", Telecommunication Systems, vol. 21:2-4, pp. 173-204.

[7] Neuts, M. F., 1981, Matrix Geometric Solutions in Stochastic Models: An algorithmic Approach. John Hopkins University, University Press.

[8] Ramaswami, V., 2000, Algorithmic analysis of stochastic models: The changing face of mathematics, Ramanujam Endowment Lecture at Anna University, Chennai, India.

[9] Riska, A., and Smirni, E., 2002, "Mamsolver: A matix analytical methos tool", in the Proceedings of the 12-th International Conference on Modeling Techniques and Tools for Computer and Communication Systems Performance Evaluation, vol. LNCS 2324, pp.205-211.

[10] Roberts, J. W., 1981, "A Service System with Heterogenous User Requirements", Performance of Data Communication Systems and their Applications, pp. 423-431.

[11] Ross, K. W., 1995, Multiservice Loss Models for Broadband Telecommunication Networks. New York: Springer-Verlag.