Performance Evaluation and Modeling of Web Server Systems

M. Riaz Moghal, N. Hussain, M. S. Mirza, M. Waris Jarral, M. S. Choudry
Department of Electrical Engineering,
University College of Engineering and Technology,
Mirpur (AJK)-10250, PAKISTAN

Abstract: - Regardless of the rising number of WWW Web servers in use everyday, not much is explored about their performance attributes. We present a high-level, open queuing network model from which we obtain several general performance results for web servers on the Internet. We also evaluate multiple-server systems. A hypothetical upper limit on the web server capacity is also worked out. As Web servers reach this limit, response time increases abruptly towards infinity, which disables the server. Putting constraint on the server's parallel connections prevents this problem. The effects of request (i.e., file) size, web server speed, and network bandwidth on response time are also explored. In addition, the relative advantages of several methods of improving server performance are investigated.
Key-Words:- Web Server, Load Balancing, Queuing Model Theory, Performance Analysis, Load Sharing, Resource Management.

1 Introduction

The World Wide Web (WWW) has gained exceptional boost in the past few years. Millions are browsing the Web and hundreds of new Web sites are added everyday [12]. Regardless of the rising number of Web servers in use, not much is known about their performance attributes. Web server vendors are happy to praise the performance virtues of their products, and industry experts abound with theories about how to serve data faster; but these virtues and theories are generally based upon unreliable evidence, or narrow empirical evidence that has little general utility. Web server hardware, software, and internet connectivity are required elements of any web site, and they are all costly. For the best possible performance for any web site, a perception of the interrelated effects of these three elements on web server performance is significant.

We present an analytical performance model of web servers in which the web server and the Internet are modeled as an open queuing network in an integrated way. We achieved interesting results during the analysis of the model. For example, as the load on the web server increases the time required to service a file increases very gradually (almost very slightly) up to a point; thereafter, it increases abruptly and asymptotically toward infinity. This asymptote defines an upper limit on the serving capacity of web servers. This limitation is especially sensitive to the average size of the files served. As the load on the server approaches the limit, a slight increase in the load can suddenly thrust the server into a deadlock situation, where it attempts to serve more and more files at slower and slower speeds such that almost no files are effectively served. Most of the UNIX based web servers allows a large number of concurrent connections, and is especially prone to this problem. Ironically, it is the servers on Windows and Macintosh platforms, often criticized for their limited number of concurrent connections, which are guaranteed to prevent such server deadlock. The queuing model was also used to explore multiple-server systems. Results show that in a multiple-server system, service load balancing among the servers is vital to optimal performance. Indeed, a two-server system in which one server is slower than another appears to perform worse than the faster server alone.

Several common techniques for improving web server performance were investigated and compared. As predicted, when the network speed is the bottleneck, increasing network speed generates the best performance improvement. But when the bottleneck is the server itself, the best choice depends on several issues. The configuration of the remaining paper is as follow. Section 2 presents related work. Section 3 discusses queuing theory. Section 4 presents web server model. Section 5 presents analysis of web server model. Section 6 concludes papers. Finally list of references is presented.

2 Related Work

A web browser program and web server comprises a client- server system. A web server is just like a file server linked to its clients via the Internet. Researchers have used queuing models to analyze client-server systems [1-6, 9-11], but not all of these explorations analyze the performance of these systems, [only 1-5 focus on performance] focusing instead on fault tolerance [10] or file storage characteristics [1]. The structure of the WWW web, the heterogeneous nature of web clients, and the unconventional behavior of the HTTP protocol cause these models miserable as models of web servers. The queuing models that have been investigated in the references are all closed network models. These investigations predate the appearance of the web as an entity valuable of exploration. These models are designed with traditional file servers in mind, and assume a identical LAN network architecture where the clients are both constrained and well defined. In these circumstances a closed queuing network model is most suitable. Web servers do not match to the assumptions built into earlier models. For any given web server the number of prospective clients is in the tens of millions [12], and consist of a variety of different web browsers running on diverse hardware platforms and connected to the Internet at numerous different speeds [7, 8]. Hence, the web does not correspond to a closed queuing system.

3 Queuing Model Theory

Web servers typically process many concurrent jobs (i.e., file requests), each of which competes for different shared resources: CPU time, file access, and Network Bandwidth. As only one job may use a resource at any time, all other jobs must wait in a queue for their turn at the resource. As jobs receive service at the resource, they are removed from the queue; new jobs arrive and join the queue. Queuing theory is a tool that helps to calculate the size of those queues and the time that jobs spend in them. The present work focuses on the number of concurrent HTTP GET file requests managed by a server, and the total time required to service a request.

Now we present very important concepts from queuing theory. Complete information is presented elsewhere [2, 3, 4, 5, 9]. In Queuing theory every service or resource is an theoretical system consisting of a single queue feeding one or more servers. Every queue has an arrival rate (A_r) -- the average rate at which new jobs arrive at the queue. Average amount of time that it takes a server to process such jobs is the service time (S_t) of the server, and the average amount of time a job spends in the queue is the queuing time (Q_t). The average response time (R_t) is simply $S_t + Q_t$. If the arrival rate (A_r) is less than the service rate ($1/S_t$) then the queuing system is said to be stable; all jobs will eventually be serviced, and the average queue size is bounded. On the other hand, if $A_r > (1/S_t)$ then the system is unstable and the queue will grow without bound. The product of the A_r *and* S_t yields the server utilization $Su = Ar * St$
Server utilization is number between 0 and 1 for all stable systems. A utilization of 0 represents an idle server, while a utilization of 1 represents a server being used at maximum capacity. If inter-arrival time between jobs, is defined as ($1/A_r$), is random and unpredictable then the arrivals show an exponential or "memory less" distribution. This distribution is particularly important to queuing theory. A queue in which the inter-arrival times and the service times are exponentially distributed is known as an M/M/C queue, where the M's denote the Markov or memory less nature of the arrival and service rates, and the C represents the number of servers attached to the queue. When service history of a queuing system is irrelevant to its future behavior --only the current state of the system is important-- that history can be ignored, which greatly simplifying the mathematics. For example, the response time of an M/M/1 queue is simply $Rt = \frac{St}{(1 - Su)}$

The response time curve of an M/M/1 queue as a function of server utilization is shown in Figure 1. At a server utilization of 0 the response time is just the service time; no job has to wait in a queue. As server utilization increases, the response time of the queue increases gradually. Only when the server utilization reaches to 1, the response time climb sharply toward infinity. As we will demonstrate below, web servers behave similarly. Little's Law ($N = Ar*Rt$) states that the average number of jobs waiting in the queue (N) is equal to the product of the average arrival rate and the average response time. Little's Law is general, and applies to all queuing systems that are both stable and conservative (i.e., no work is lost when switching between jobs). Little's Law is particularly useful when applied to queuing networks.

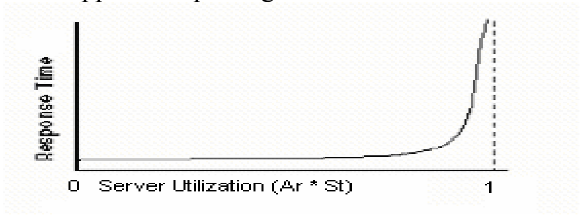

Figure 1: Response Time(Rt) of Queuing System

4 Web Server Model

We present a high level view of a web server, modeled as an open queuing network. Our objective is to construct a generally appropriate model that abstracts all hardware and software details, but is detailed enough to produce significant performance results about the relationship between server and network speeds. In the present model we have not considered the low-level details of the HTTP and TCP/IP protocols. Similarly, the present model acts as a simple file server over the Internet, and ignores both "if-modified" behavior and common gateway interface applications (CGI) [12]. A typical web server queuing network model is presented in Fig. 2.

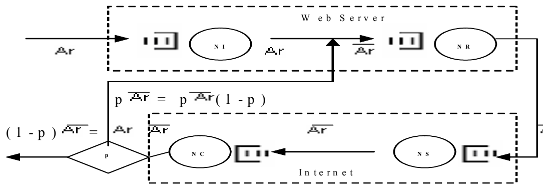

Figure 2. Web Server Queuing Network Model

The network consists of four nodes (i.e., single-server queues); two for modeling the Web server itself, and for two modeling the Internet communication network. Jobs (File requests) arrive at the Web server at Arrival rate *Ar*. All single-time "initialization" processing is performed at node N_I. The File request then proceeds to node N_R where a single buffer's worth of data is read from the file, processed, and passed on to the network. At node N_S this data block is transmitted to the Internet at the server's transfer rate (e.g., 1.5 MBits on a T1 line). This data moves via the Internet and is received by the client's browser, denoted by node N_C. If the file has not been fully transmitted, the "job" branches and returns back to node N_R for further processing. Otherwise, the job is complete, and exits the network.

Here the branch is a probabilistic one; given an average file size of F_S and buffer size B_S, the probability that the file has been fully transmitted is p = B_S /F_S . In addition to, the arrival rate at node N_R (A_r') is the sum of the network's arrival rate (A_r), and the rate of the jobs flowing from N_C back to N_R. Numerous simplifying suppositions are made into the model. The effect of the HTTP GET requests on the network is not considered, since the requests are normally much smaller than the files that are served. It is also, supposed that the size of requested files (and thus the service times) are distributed exponentially. Although this may not be true for some web sites, this supposition is conservative; values based on conservative approximations denote an upper limit on the true values. Also, given fixed size buffers the service rates at nodes N_S and N_C are probably not exponential. Again, this is a conservative approximation. As the set up best fits Jackson's Network model, we will treat this model as Jackson's Network Model too and so the response time of the queue is given by

$$R_T = \frac{F_S}{C_{NB}} + \frac{I_T}{1 - A_r I_T} + \frac{F_S}{S_{NB} - A_r F_S} + \frac{F_S(B_S + D_{SR} S_{ST})}{B_S D_{SR} - A_r F_S (B_S + D_{SR} S_{ST})}$$

where the parameters in the formula are: i) Network Arrival Rate (*Ar*) ii)Average File Size (F_S), iii) Buffer Size (B_S) iv) Initialization Time (I_T), v) Static Server Time (S_{ST}) vi) Dynamic Server Rate (D_{SR}), vii)Server Network Bandwidth(S_{NB}), viii)Client Network Bandwidth (C_{NB}).

Before analyzing the model, it is important to understand the definition of the above model parameters, and how they were applied during the analysis presented below. Network arrival rate (*Ar*) is the average number of HTTP file requests (i.e., "hits") received by the web server each second. It is significant to realize that *Ar* represents an average and not an instantaneous value. It is easier to calculate a corresponding "hits per day" value; just multiply *Ar* by 60*60*24 = 86,400. Figure 3 illustrates this correspondence between arrival rate and "hits per day".

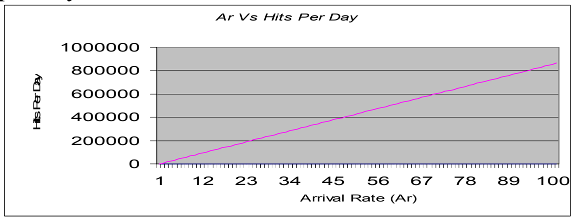

Figure 3. Arrival Rate Vs. Hits Per Day

Fs is an average File size (in bytes) of the files served. Apparently, this value will vary usually from one web site to another. On the other hand, after visiting 1000 web pages at random --using the "random link" feature available from several search engines-- and observing the size of every file received, including graphics, the average thus obtained was 5,275 bytes. This value has been used to produce some of the graphs below, wherever a fixed value of *Fs* was required. *Bs* is the buffer size of the file chunks that are sent from the server across the Internet to the client's browser. Mostly, this value represents the disk block size of the server machine. Analysis of our model demonstrates that this value plays a minor role on overall server performance. An arbitrary value of 2000 bytes was used to generate all the graphs below.

I_T is the initialization time, S_{ST} is Static server time, and D_{SR} is Dynamic Server Rate, all together explain the speed at which the web server handles web file requests. We have represented the average time required to perform various one-time initialization tasks for each job (e.g., suffix mapping). The service rate of the N_I node is 1/I_T. S_{ST} denotes the time spent processing a buffer that is independent of the size of that buffer. Finally, D_{SR} denotes the rate (bytes/second) at which the server processes the buffer. The service rate of the N_R node is 1/ [S_{ST} + (B_S /D_{SR})]. Web servers running on new potential computers can serve data much faster than today's networks can transmit it. Server network bandwidth (S_{NB}) and Client network bandwidth (C_{NB}) collectively denote the transmission speed of the Internet. S_{NB} represents the speed at which the server sends a buffer of data to the Internet. Typical values for S_{NB} are (128 Kbits/sec - ISDN, 1.5 Mbits/sec - T1, and 6 Mbits/sec - T3). C_{NB} represents the average speed at which client software receives a buffer. Calculating average the results from a current Internet user's survey [7, 8], a rational value for C_{NB} is 707 Kbits /sec; this value was used to produce the graphs below.

5 Analysis of the Server Model

The response time curves for the web server model appear like Figure 1. Figure 4 shows the response time for a typical server connected to the Internet via a T1 line. Here the values of *Ar* less than 35 (that's 3,024,000 hits/day!) the response time (R_T) is a about fraction of a second. Although as the server reaches full utilization R_T grows asymptotically toward infinity. For this Web server is having a theoretical upper bound 3,024,000 on the number of hits per day that can be serviced. We will refer to this boundary as the maximum capacity (M_C) of the web server -- M_C is also the service rate of the web server system.

This result may be counter-intuitive. It is a common myth that web servers have no maximum capacity --all jobs will eventually be serviced, although slowly--and that response time grows approximately linearly as *Ar* increases --the decay in performance is gradual. These myths could have miserable consequences if web server managers apply it. Suppose a server is easily handling X hits per day, average response times are 50% below unacceptable values, and server utilization is increasing by only 2% of X per week. Consistent with misunderstanding above it will take almost a year before server response times double. However, if the server is near maximum capacity then response times may jump well beyond acceptable levels in a single busy day. Worse, the increased response times may be so amazing that they exceed the patience of people browsing the site. At that point, the web site is experiencing a situation similar to deadlock, where it attempts to serve more and more files at slower and slower speeds such that almost no files are successfully served.

In the presence of this situation, the remainder of this paper is dedicated to answering three questions. i) How do the model parameters above influence response times and, in particular, maximum capacity? ii) How can web servers operating near maximum capacity prevent a deadlock situation? iii) What strategies most effectively increase the performance of a Web server?

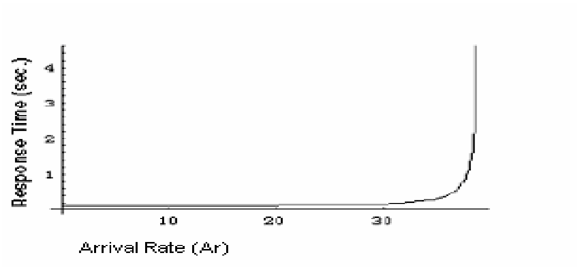

Figure 4. A Typical Response Time vs. Arrival Rate Curve

5.1 What Affects Response Time

In fact, just about everything influences response time, at least a little. But the influence of Buffer size is small, and here after buffer size is assumed to be 2000 bytes. Similarly, the Client Network Bandwidth has a very minor effect on response time and no effect on maximum capacity. Since web servers have no effect over the Client Network Bandwidth anyway, it is hereafter assumed to be 707 Kbits/second. The effects of both Initialization Time and Static Server Time can be simulated by a slight increase in Dynamic Server Rate. For this investigation it is easier to let Initialization Time and Static Server Time be 0 and to let Dynamic Server Rate alone denotes server speed. The remaining model parameters -- Average File Size, Dynamic Server Rate, and Server Network Bandwidth-- all significantly influence both response time and maximum capacity.

As the effect of Average File Size on Response Time and Maximum Capacity is always substantial, the effects of Dynamic Server Rate and Server Network Bandwidth depend upon whether the system bottleneck is the web server or the network bandwidth. Because modern computers can serve files at Ethernet speeds and beyond (10+ Mbits), and the typical Internet connections (i.e., 28.8K, ISDN, T1, T3) are slower, the network bandwidth is almost always the bottleneck. Generally only very active multi-server sites (e.g., Yahoo, etc) have enough network bandwidth that the web server becomes the bottleneck. In this situation response time and maximum capacity are measured exclusively by network speed (S_{NB}) and average file size (F_S); server speed (D_{SR}) is not significant. In those exceptional cases when the server is the bottleneck, it is server speed and average file size that are significant, and network speed, which can be neglected.

Furthermore, when the network is the bottleneck, average file size (F_S) has a major, effect on response time, as illustrated in Figure 5. This graph was generated assuming the network connection is a T1 (1.5 Mbits). The ridge represents the maximum capacity asymptote (values behind the ridge are not meaningful). Observe that for any values of average file size the shape of the (response time vs. arrival rate) response time curve is essentially the same as in Figure 1. The greatest influence of average file size is on the maximum capacity asymptote (M_C), which appears to decrease exponentially with respect to average file size.

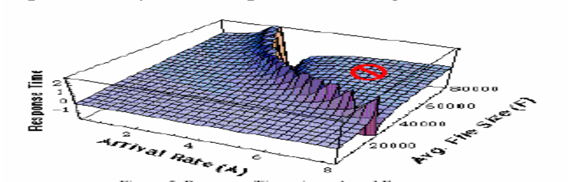

Figure 5. Response Time given A and F

It is logical that increasing Average File Size decreases maximum capacity. A web server that serves many large files uses much of the available network bandwidth to do so. However, it is somewhat surprising that this decrease is not approximately linear, gradual, and predictable. In fact, when Average File Size is relatively small (e.g., 5 Kbytes), a minor change in Average File Size can have a great effect on maximum capacity. But when Average File Size is already large, maximum capacity is already low, and small changes in Average File Size have little effect.

Figure 6 demonstrates the relationship between network bandwidth (S_{NB}) and response time. Realize that the maximum capacity ridge is straight; hence, maximum capacity increases approximately linearly with respect to network bandwidth.

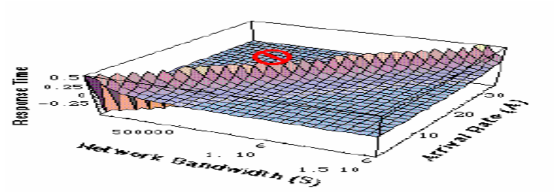

Figure 6. Response Time given A and S.

The collective influences of Average File Size and network bandwidth on maximum capacity are presented in Figure 7. Observe that the effects of Average File Size on maximum capacity are very volatile for average file sizes fewer than 20 Kbytes.

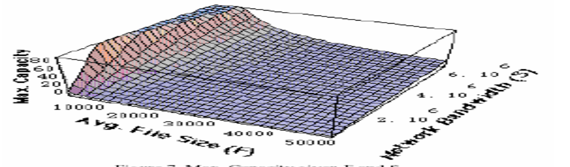

Figure 7. Max. Capacity given F and S

5.2 Preventing Deadlocks

Deadlock situations arise when new jobs are arriving nearly as fast or faster than they are being served. The only approach to prevent this situation is to stop adding jobs to the queue as *Ar* is close to M_C. It is hard for web servers, because they typically do not watch arrival rate. But according to Little's Law, the number of jobs in a queuing system (*N*) is equal to $Ar*R_T$. For web servers, *N* relates to the number of concurrent open TCP/IP connections: a known quantity.Figure 8 demonstrates the relationship between and *N* for the same server demonstrated in Figure 4. As required by Little's Law, the shape of this curve imitates that of the response time curve, and the asymptote occurs at M_C. Thus, the amount of *N* can be used to identify imminent deadlock situations.

For most Windows NT and UNIX based web servers, the number of concurrent connections (*N*) is practically unlimited. It appears that this "characteristic", often cited as the main advantage of UNIX based servers, is in fact a curse.

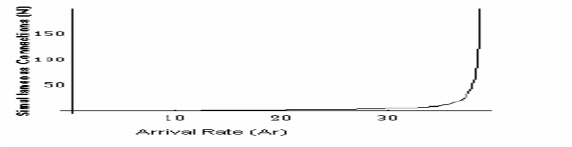

Figure 8. Concurrent Connections vs. Arrival Rate (Ar)

In contradiction, most Macintosh and Windows based web servers already have a bound on the number of concurrent open TCP/IP connections, constrained by either the operating system or the server software. When a web server is approaching maximum capacity, it should respond to new file requests with the HTTP "come back later" response, and continue to complete the jobs already in its queue. The browser software should then automatically resubmit the request after several seconds when the server is hopefully less busy. Fatefully, few servers produce this response, and no known browser supports it.

5.3 Increasing Web Server Performance

As web server performance becomes unacceptable, there are three obvious alternatives for increasing it: i) Replace the server with faster one ii) Increase Network Bandwidth iii)Add additional server.

It is already demonstrated that using a faster computer (i.e., increasing D_{SR}) or improving network bandwidth (S_{NB}) decreases response times and increases maximum capacity. What we have also done is the effect of adding additional servers. It is not always cost effective to completely replace a working computer with a faster model. Instead, it is common practice to add additional computers. The web site content is then either mirrored on all server machines, creating a RAIC (Redundant Array of Inexpensive Computers), or divided between the server machines [12]. This scalability is an appealing feature of web servers that run on relatively inexpensive machines.To workout the efficiency of a multi-server system the queuing network model was altered as presented in Figure 9. Jobs are directed to node N_{R1} with probability q, and to N_{R2} with probability (1 - q).

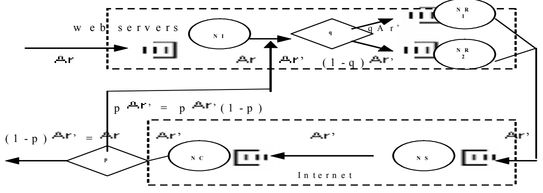

Figure9.Multiple-Servers Queuing Network Model

With this new model we have explored the relative advantages of RAICs. The solid curve in Figure 10 illustrates response times of a web server in the region well below maximum capacity (Dynamic Server Rate = 10 Mbits (Ethernet), network bandwidth = 1.5 Mbits (T1), and Average File Size = 5000). Four alternatives are also investigated.Apparently, the best option in this situation, when the network bandwidth is the bottleneck, is to increase the network bandwidth. Doubling the server speed demonstrated a very small improvement. Adding a second identical server in a RAIC (not shown) had no effect at all. Finally, adding a second, but slower, server in a RAIC actually increased the response time (i.e., decreased performance).

Multiple-server systems are very sensitive to mismatched loads. Mismatched servers in a RAIC (i.e., q [not equal] 0.5) overload the slower server while the faster server may be idle. In non-RAIC, multiple-server systems mismatched loads can also be caused by different average file sizes (F_S). Though, this can be exploited to help balance the load between different model server machines.

Increasing performance is even more interesting for those very active sites where the server itself is the bottleneck. Figure 11 shows this situation. The best option, as expected, is to double the server speed. The next best option depends upon the arrival rate experienced by the site. For arrival rates under 110 (that's 9,504,000 hits per day!) the second best choice is to double the network bandwidth.

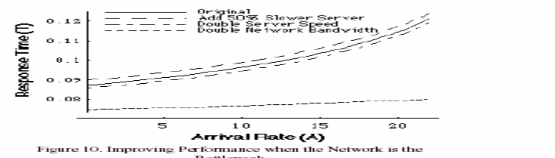

Figure 10. Improving Performance when the Network is the Bottleneck

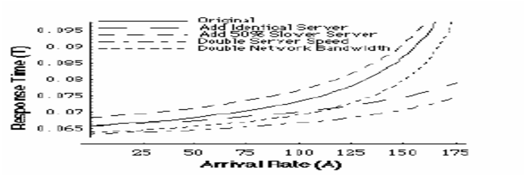

Figure 11. Improving Performance when the Server is the Bottleneck

However for higher arrival rates the second best choice is to add another identical server in a RAIC. Finally, the worst option is to add a slower server in a RAIC, which causes a decrease in performance.

5 Conclusion

In the paper we have presented an abstract performance model of web servers in which the web server and the Internet are collectively modeled as an open queuing network. Analysis of this model produces several interesting results. Most significantly, as the service load on a Web server increases the time required to serve a file increases very gradually (almost slightly) up to a point; thereafter, it increases suddenly and asymptotically toward infinity. This asymptote denotes a clear upper limit on the serving capacity of web servers. This maximum capacity limitation is mostly sensitive to the average size of the files served. By bounding the number of concurrent connections, a web server can prevent deadlock situations that occur as the server load proceeds maximum capacity. Also, the relative merits of various methods for improving web server performance were analyzed.

References:

[1] King, P. J. B., Computer and Communication Systems Performance Modeling, Prentice Hall International, UK,1990
[2] Kleinrock, L., Queueing Systems, Volume I: Theory, John Wiley and Sons, New York, 1976.
[3] Drakopoulos, E. and M. J. Merges, "Performance Analysis of Client-Server Storage Systems", IEEE Transactions on Computers, Vol. 41, No. 11, November 1992.
[4] Gelenbe, E. and I. Mitrani, Analysis and Synthesis of Computer Systems, Academic Press, New York, 1980.
[5] Kleinrock, L., Queueing Systems, Volume II: Computer Applications, John Wiley and Sons, New York, 1976.
[6] Petriu, D. C. and C. M. Woodside, "Approximate MVA from Markov Model of Software Client/Server Systems",1991
[7] Pitkow, James E. and Colleen M. Kehoe, Results from the Third WWW User Survey, The World Wide Web Journal, Vol. 1, No. 1, 1995.
[8] Pitkow, James E. and Colleen M. Kehoe, "The Fourth GVU CenterWWW,User,Survey",.www.cc.gatech.edu/gvu/user_sur
[9] Pujolle, G. and E. Gelenbe, Introduction to Queueing Networks, John Wiley & Sons, New York, 1987.
[10] Puliafito, A., S. Riccobene, and M. Scarpa, "Modeling of Client-Server Systems", 1995.
[11] Trivedi, K. S., O.C., Ibe, and H. Choi, "Performance Evaluation of Client-Server Systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 4, N. 11, Nov. 1993
[12] Wiederspan, J. and C. Shotton, Planning and Managing Web Sites on the Macintosh, Addison-Wesley, 1996.