

Modeling of Session Persistence in Web Server Systems

Wei LI and Rein VESILO
Department of Electronics
Macquarie University, NSW, 2109
AUSTRALIA

Abstract—Modern web server systems use multi-tier architectures with load balancing devices to increase system capacity, response times and system reliability. Session persistence ensures that requests from the same session are directed to the same server. This paper examines techniques for modeling web server systems in the presence of load balancing and session persistence. The join-the-shortest (JSQ) queue load balancing technique is used for load balancing. First, a model of a single session is developed that can be used to set session persistence timers. Next, an infinite server model is used to determine session occupancy and session imbalances across multiple servers. This is used for more detailed modeling of the multi-tier system as a closed queueing network using an approximate Mean Value Algorithm (MVA) that includes non-exponential service times. It can be used to determine mean response times, throughput and processor utilisation.

Keywords: *Web server systems, session persistence, join the shortest queue, infinite server, mean value algorithm*

I. INTRODUCTION

As the use of World Wide Web services has grown, users have come to expect low site downtime and short response times. Modern web sites are based on a multi-tier architecture. In general, they consist of a web layer where web servers with Internet connections are located, an application layer that contains business logic and a database layer that contains customer and business information. The use of load balancers to distribute network load among clustered servers in tiers, increases web site capacity, reduces download times and improves reliability. Load balancers make decisions regarding which server is “best” to assign a new connection to. Common load balancing policies include, Simple Round-Robin, Weighted Round-Robin, Simple Least Connections and Weighted Least Connections. Persistence policy ensures that the same user is connected to the same server for the duration of the session. This is important in applications such as web shopping. Without session persistence, the shopping cart information would be scattered over the pool of servers. System designers are interested in how to design web systems where load balancing is used and how to provide enough capacity to meet response time and utilisation targets. This requires suitable session models, knowledge of session statistics and techniques for analysing models. We consider session modeling at two levels: within a session and in the context of the entire web site. At the session level, a session

consists of a sequence of requests. When the time between requests, determined by a session persistence timer, is larger than a threshold the session is terminated. If the session timer is too short a genuine session might be terminated prematurely. If the timer is too long then resources might be held by the session too long. At the web site level, modeling has to include many features such as the presence of caches, load sharing and load balancing, job migration, replication of servers, database locks, concurrency limits and admission controls. In addition, the sizes of requests and lengths of session can exhibit high variability including heavy-tailed statistics. In this paper, we focus on modeling the load sharing and load balancing aspects of the system and concentrate on the join-the-shortest queue (JSQ) (least connections) method of load sharing.

We consider two main questions. First, how can we model a session to determine suitable values for the session persistence timer? Secondly, how can we model a multi-tier web site to take into consideration the effects of session persistence on load sharing arrangements? To date, there has been little research on the first question. Often, there are only broad guidelines given by equipment manufacturers. Regarding the second question, models of web sites have been developed using queueing networks. Slothouber [10] developed a model of a single tier web site delivering static content. Menasce [1] has developed a queueing network model of a multi-tier site. Urgaonkar et al. [9] developed a model that includes concurrency limits in tiers. Our contributions are as follows. In answer to the first question we develop session level models that can be used to estimate probabilities of premature session termination and set persistence timers. Regarding the second question, we develop a JSQ model for infinite server stations. Each station represents a single real server and the model can be used to determine the probability of how many sessions are in progress at a particular server. We then apply the results of the JSQ infinite server model to model a multi-tier system as a closed queueing network. This model we use is similar to the one developed by [9] but the load distribution parameters have been determined from the JSQ model and the model has been extended to include non-exponential session times. We allow them to be hyperexponential and solve the network using an approximate mean value algorithm (MVA). We present an example of a two tier system obtaining mean response times, processor utilisation and throughput. In the remainder of the paper, Section II overviews multi-tier architectures, session persistence and session statistics; Section III presents the intra-

session model; Section IV presents the infinite server model; Section V presents the multi-tier model and approximate MVA algorithm; and, Section VI is the Conclusion.

II. MULTI-TIER ARCHITECTURES

A typical multi-tier system is shown in Figure 1. The front-end consists of web and application services accessible by the users over the Internet. The back-end consists of database servers and other components like firewalls. The web servers either deliver the content requested by clients or pass the request to the application server tier. If the request involves data stored in the database, the application tier sends a query to the database tier. Early load sharing implementations used round robin DNS and proprietary solutions as used in early Netscape Navigator versions. Later, load balancing technology used only up to Layer 4 (L4) switching. L4 solutions still assume servers provide static content which can be a problem with dynamic content generation. At the TCP connection level set-up times occur due to the TCP Slow Start mechanism. Layer 7 (L7) load balancing uses application information, examining packet content beyond Layer 4 headers. L7 load balancers can use source IP addresses, source ports, cookie techniques such as cookie-read, cookie-insert, and cookie-rewrite and SSL session ids, used in SSL sessions (HTTPS). Set-up delays can occur due to setting up SSL devices and storing initial cookie information.

Customer behavior models characterize session level user behavior. [1] uses a Customer Behavior Model Graph (CBMG) that is a state machine with states corresponding to different user activities such as Browse, Search, Select, Add to Cart, Pay. There is an entry state, home page and exit point. Users move between states as they visit different pages on the web site. The TPC-W is a transactional web e-commerce benchmark introduced by the Transaction Processing Performance Council [8] and uses a CBMG type model. It classifies interactions into two broad categories. (1) Browse interactions involve browsing and searching but no product ordering. States of the CBMG that fall in this category are Home, Browse, Select, Product Detail, and Search. (2) Order interactions involve product ordering only and include: Shopping Cart, Login, Buy Request, and Buy Confirm. TPC-W specifies three different types of sessions according to the mix of browse and order interactions found in each session: Browsing mix: 95% browse and 5% order; Shopping mix: 80% browse and 20% order; Ordering mix: 50% browse and 50% order. These models are useful in determining the probability of moving from one tier to another in the multi-tier model. Browsing involves mostly static pages whereas ordering involves more database interaction.

Measurement of file sizes in the Web [5] has shown evidence of heavy-tailed distributions. Both the Pareto model ([5], [6]) and lognormal model ([5]) have been fitted to the distribution of file sizes on typical web servers. Our models will not use these sorts of heavy tailed distribution but do include non-exponential distributions with more variability than exponential distributions. When modeling arrival times of sessions and requests some researchers also found evidence of heavy tailed interarrival times but others found that the Poisson approximation was satisfactory ([6]).

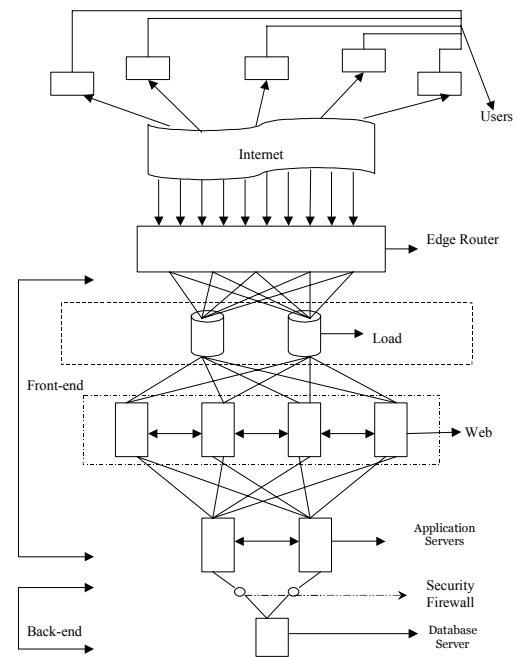


Figure 1. A typical multi-tier web system

III. SESSION PERSISTENCE TIMER MODELS

A session can be defined as all requests from a client to a server with the time between requests less than a timeout value. Requests are separated by user think times. We examine models for evaluating the effect of session persistence timers. Consider a session consisting of N requests (Figure 2). Request k in the session is followed by an idle time I_k , provided it is not the last request in the session. For $N > 1$, the session is terminated the first time I_k exceeds the time out value τ_k . Let this occur for index K . A session is properly terminated if $N=1$ or if $N > 1$ and $K \geq N$. A session is prematurely terminated if $N > 1$ and $K < N$. The probability of premature termination, p_T , is:

$$p_T = P(N \geq 2 \text{ and } K < N) = \sum_{n=2}^{\infty} P(K < n)P(N = n) \\ = \sum_{n=2}^{\infty} \sum_{k=1}^{n-1} P(K = k)P(N = n).$$

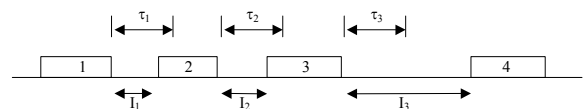


Figure 2. Sequence of session requests with persistence timers.

We assume the pairs (I_k, τ_k) are i.i.d. Let $q = P(I_k \leq \tau_k)$ be the probability the session timer does not expire. Using the geometric trials argument $P(K = k) = q^{k-1}(1-q)$ gives:

$$p_T = \sum_{n=2}^{\infty} (1 - q^{n-1})f(n) = 1 - \frac{1}{q} \sum_{n=1}^{\infty} q^n f(n) = 1 - \frac{1}{q} F(q)$$

where $f(n) = P(N=n)$ and $F(q)$ is the probability generating function of $f(n)$. If the number of requests in a session is

geometrically distributed then $f(n) = p^{n-1}(1-p)$ where p is the probability of one more request after the current request. In that case, $F(q) = q(1-p)/(1-pq)$ giving

$$p_T = 1 - \frac{1-p}{1-pq} = \frac{p(1-q)}{1-pq}.$$

As q approaches 1 then p_T approaches 0 and as q approaches 0 then p_T approaches p . The mean number of requests in a session is $1/(1-p)$. Regardless of the idle time and timer distributions the probability of premature session termination can be found. If idle times are exponentially distributed with mean a and τ_k is a fixed value τ then $q = 1 - \exp(-a\tau)$. Knowing a we can choose τ to meet a given target for p_T .

A. Multiple sessions access a shared resource

We can also model resources accessed by multiple sessions. Assume a set-up time is required to activate a resource if it has been left idle. This can model a file cache where files are shared by different sessions; for example, catalogue files. Assume requests arrive according to a Poisson stream with rate λ and that processing times have a general probability distribution. This is an M/G/1 queue with set-up time and is closely related to an M/G/1 queue with exceptional first service times since the exception first service time = set-up time + normal service time. This system is analysed in [7] using Laplace-Stieltjes transforms (LST) from which the mean waiting time $E(W)$ and mean sojourn time $E(S)$ can be determined. If $E(B_n)$ the mean service time of a normal customer, $E(B_f)$ the mean service time of an exceptional customer, $E(R_n)$ the mean residual service time of the normal customer and $E(R_f)$ the mean service time of an exceptional customer then

$$E(R_f) = E(B_f^2)/2E(B_f) \quad \text{and}$$

$$E(R_n) = E(B_n^2)/2E(B_n) \quad \text{and the mean sojourn time is}$$

$$E(S) = E(W) + E(B) = E(W) + q_f E(B_f) + (1-q_f)E(B_n)$$

$$\text{where } q_f = (1 - E(B_n)\lambda)/(1 + \lambda E(B_f) - \lambda E(B_n)).$$

This model can be extended to a shared resource with a persistence timer. If the resource has no requests to process and becomes idle then as long as new request is received before the timer expiration the resource can be activated without a new setup. [7] examines a generalized versions of this in a queue with N-policy with setup and closedown time. (N-policy operates so that after the server becomes idle and the closedown time has expired the server will only become active when queue size reaches N). We have $N=1$. If $E(S)$ is the mean setup time, $D^*(s)$ is the LST of the closeddown time, ([7]):

$$E[W] = \frac{\lambda E(B_n^2)}{2(1-\rho)} + \frac{D^*(\lambda)\{2\lambda E[S] + \lambda^2 E[S^2]\}}{2\lambda\{D^*(\lambda)(1 + \lambda E[S]) + 1 - D^*(\lambda)\}}$$

where $D^*(\lambda)$ denotes the LST of closeddown time. For constant closeddown times with value a and $D^*(\lambda) = e^{-\lambda a}$.

IV. SESSION LEVEL MODELLING

In session level modeling we only use the knowledge of session times and arrival rates to examine the system. Assume sessions arrive according to a Poisson distribution with rate λ and session times are i.i.d with mean session duration μ . We

assume session times are not affected by each other and can model the aggregate system as an infinite server M/G/ ∞ queue. The probability distribution for number of sessions in the system is a Poisson distribution with mean λ/μ .

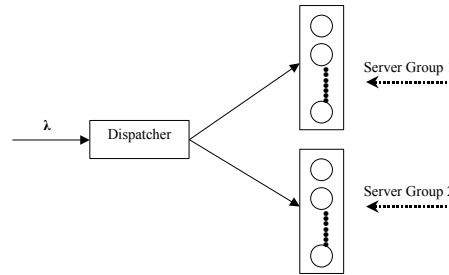


Figure 3. Infinite Server JSQ model

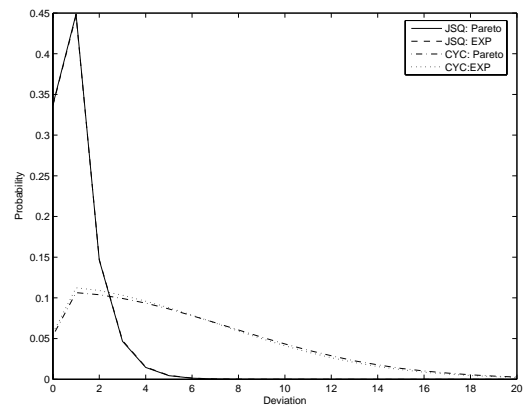


Figure 4. The probability of load imbalance for JSQ and CYCLIC policies

We now model the affect of load balancing and use the JSQ scheduling policy, where a newly arriving session is dispatched to the infinite server with the fewest waiting jobs (Figure 3). In contrast to JSQ for FCFS, there has been limited research done on the JSQ for infinite server (IS) stations. Previous work includes [2] and [3]. In comparison, JSQ for FCFS (first-come-first served) stations has received quite a bit of attention and a number of optimality results are available (see [4]). In the case of two parallel M/M/ ∞ queues assume all servers in the first queue work at service rate μ_1 and all servers in the second queue work at service rate μ_2 . The JSQ-IS system can be modeled as a two-dimensional Markov chain, whose state may be represented as the tuple (n_1, n_2) . In the symmetric case, we make $\mu_1 = \mu_2$. The balance equations for this system can be obtained and are given in [3] and it gives the most detailed analysis of this system that we know of. It gives a number of solutions that include an exact solution and a number of numerical and asymptotic methods. We used simulations and validated the results using [3]. We simulated both Pareto (index = 1.5) and exponential service times using both JSQ and cyclic allocation. We let the arrival rate be $\lambda = 4 \text{ s}^{-1}$ and the mean service time be 25 s giving a mean number in the system of 100. We plotted the probability of load imbalance in the system, defined by $D = |n_1 - n_2|$. The results are shown in Figure 4. The JSQ curves are clearly better than the Cyclic curves. We found that having an exponential or Pareto

distribution did not matter much. We also found in other studies that the index of the Pareto distribution did not matter much either.

V. MODELLING MULTI-TIER SYSTEMS

We modeled the web server system as a queueing network and used a similar multi-tier model as [9] (Figure 5). Tier 0 represents user think times and is an infinite server node. There are M tiers in the web site: Tier 1, Tier 2, ..., Tier M . Tier N contains K_N servers. For simplicity, we assume FCFS servers in these tiers. After leaving Tier i , a request either returns to tier $i-1$ with probability p_i or proceeds to tier $i+1$ with probability $(1-p_i)$. Note in Tier M , $p_M=1$.

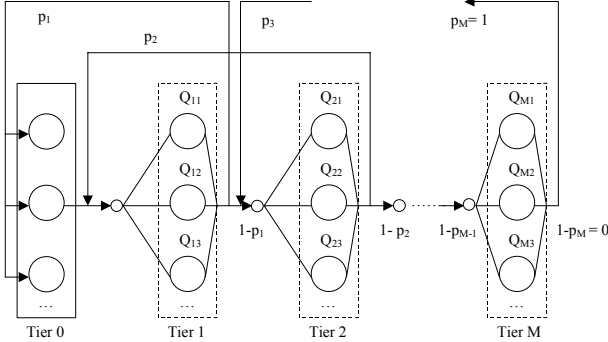


Figure 5. Model of M-tier Internet web site

We used the Mean Value Analysis (MVA) algorithm to compute performance quantities such as mean waiting time, throughput, and the mean number of jobs at each node. The MVA algorithm for closed queueing networks was developed by [11] and can compute performance measures without computing the normalization constant. Suppose the number of nodes is N and that there are K requests in the network. Let the arrival rate at node i be λ_i . The visit ratio is the mean fraction of visits (v_i) of a request to node i relative to a reference node (usually node 1) with $v_i = \lambda_i / \lambda_1$. We define throughput $\lambda = \lambda_1$. To find the visit ratios for nodes the traffic equations $v_i = \sum_{j=1}^N v_j p_{ji}$ are solved where p_{ji} are the routing probabilities that a job goes to node i after leaving node j . Let μ_i be the mean service time, $T_i(K)$ the mean response time, $Q_i(K)$ the mean number of requests and ρ_i the utilisation at node i when there are K requests in the network. The MVA algorithm involves iteratively solving three equations:

1. $T_i(K) = [1 + Q_i(K-1)] / \mu_i$. (node response time)

This equation is based on the result that the probability distribution of the number of jobs seen by an arrival to a node with K jobs in the network is the same as that when there is one less job ($K-1$) in the network. For IS nodes $T_i(K) = 1 / \mu_i$.

2. $\lambda(K) = K / \sum_{i=1}^N v_i \times T_i(K)$ (overall throughput)

3. $Q_i(K) = \lambda(K) \times T_i(K) \times v_i$, (node queue length)

Equations 2 and 3 are based on Little's Law. The Bard-Schweitzer approximation [12] of the MVA estimates $Q_i(K-1)$ by $Q_i(K-1) = (K-1)Q_i(K) / K$. [13] presents

various methods for analysing queueing networks with non-exponential service times. We used three of these techniques. In the first (AMVA1), compute T_i using:

$$T_i = \tau_i \left(1 + \frac{K-1}{K} (K - \rho_i) \right) + \frac{K-1}{K} \rho_i L_i$$

where $L_i = \tau_i (1 + C_i^2) / 2$ is the mean equilibrium service time and $C_i^2 = \text{variance} / \text{mean}^2$ is the squared coefficient of variation of the service times. In the second (AMVA2), an interpolation for the mean residual service is used where L_i' is used instead of L_i in the previous equation:

$$L_i' = \frac{L_i}{L_i + T_i^0} \tau_i + \frac{T_i^0}{L_i + T_i^0} L_i$$

where T_i^0 is the mean time for a request to enter node i and then return back. The third method (AMVA3) is used for networks that have servers with hyperexponential service times. Such a random variable is either an exponentially distributed random variable with mean τ_S chosen with probability p_S (S =small) or an exponential distributed random variable with mean τ_L (L =large) chosen with probability $1-p_S$. The mean of the random variable is $(1-p)\tau_L + p\tau_S$ and the

coefficient of variation is $C^2 = 2 \frac{(1-p)\tau_L^2 + p\tau_S^2}{((1-p)\tau_L + p\tau_S)^2}$. The

approximate MVA algorithm then computes:

$$T_i = p_{Si} T_{Si} + (1 - p_{Si}) T_{Li}$$

$$T_{Si} = \tau_{Si} \left(1 + \frac{K-1}{K} Q_{Si} \right), \quad T_{Li} = \tau_{Li} \left(1 + \frac{K-1}{K} Q_{Li} \right)$$

$$Q_{Li} = K \frac{T_{Li}}{T_{Li} + T_i^0 - T_i}, \quad Q_{Si} = K \frac{T_{Si}}{T_{Si} + T_i^0 - T_i}.$$

AMVA1 and AMVA2 are limited to only one high variability server in a network unless decomposition methods are used. AMVA3 can be used with multiple high variability servers without needing to use decomposition methods.

A. Two-tier example

We analysed a two-tier system. The first tier had three web/application servers with requests distributed equally. The second tier had two database servers and we allowed requests to be unevenly distributed. At the session level we determined the probability distribution of sessions using a single M/G/ ∞ queue. With an arrival rate of sessions of $\lambda=4$ sessions/sec and mean session duration of 25 seconds the mean number of sessions in the system was 100 sessions. The probability of more than 121 sessions in the system was less than 2%. We choose 100 and 121 sessions to analyse. We estimated the probability of various imbalances in the second tier by simulating the JSQ infinite server model with two servers. From these we let the proportion of requests going to the two back-end servers vary with proportions (0.5,0.5), (0.45,0.55), (0.4,0.6) and (0.35,0.75). From Figure 4, imbalances up to (0.45,0.55) could be expected for JSQ but up to (0.4,0.6) for Cyclic. We assume that service times in the first tier were exponential. In the second tier service times were

hyperexponential with $p_s = 0.99$ and $C^2 = 10$. We chose probability p_1 to be 0.7 and user think times were exponential with mean 5 sec. The mean service times of all servers were adjusted proportionally to vary system capacity. We assumed that the mean service time of a back server was 5 times that of a front server. Results are shown in Figures 6 and 7. Each graph shows four curves - one, for each proportion of back tier request distribution. We present results for the AMVA3 algorithm. For $C^2 = 10$, AMVA1 and AMVA2 gave similar results. Figures 6 and 7 show that increasing imbalance causes increased response times. 121 sessions has higher response times than 100 sessions. We also tried higher values of C^2 (50 and 100) but found that the AMVA3 gave inconsistent results probably caused by the limitations of AMVA3 for high C^2 values. Tables I, II and III show performance figures for individual nodes. Node 1 models users, nodes 2,3,4 the front servers and nodes 5 and 6 the back servers. $Toth_i$ is the time for a request to leave node i traverse the rest of the network and return. The time for the user to receive a response is $Toth_1$. Increasing session numbers increased response times in all servers. The affect of imbalances on nodes 5 and 6 are seen in Table II.

TABLE I. K=100, $C^2=10$, LB2=(0.5,0.5)

Node(i)	μ_i	T_i	$Toth_i$	Q_i	ρ_i	λ_i
1	5.0000	5.0000	0.9210	84.4448	84.4448	16.8890
2	0.0200	0.0238	12.4104	0.1913	0.1608	8.0424
3	0.0200	0.0238	12.4104	0.1913	0.1608	8.0424
4	0.0200	0.0238	12.4104	0.1913	0.1608	8.0424
5	0.1000	2.0698	25.5617	7.4906	0.3619	3.6191
6	0.1000	2.0698	25.5617	7.4906	0.3619	3.6191

TABLE II. K=100, $C^2=10$, LB2=(0.4,0.6)

Node(i)	μ_i	T_i	$Toth_i$	Q_i	ρ_i	λ_i
1	5.0000	5.0000	0.9224	84.4245	84.4245	16.8849
2	0.0200	0.0238	12.4134	0.1913	0.1608	8.0404
3	0.0200	0.0238	12.4134	0.1913	0.1608	8.0404
4	0.0200	0.0238	12.4134	0.1913	0.1608	8.0404
5	0.1000	1.9912	32.5564	5.7637	0.2895	2.8946
6	0.1000	2.1277	20.9041	9.2380	0.4342	4.3418

TABLE III. K=121, $C^2=10$, LB2=(0.5,0.5)

Node(i)	μ_i	T_i	$Toth_i$	Q_i	ρ_i	λ_i
1	5.0000	5.0000	1.1232	98.8045	98.8045	19.7609
2	0.0200	0.0246	12.8341	0.2314	0.1882	9.4100
3	0.0200	0.0246	12.8341	0.2314	0.1882	9.4100
4	0.0200	0.0246	12.8341	0.2314	0.1882	9.4100
5	0.1000	2.5388	26.0361	10.7507	0.4234	4.2345
6	0.1000	2.5388	26.0361	10.7507	0.4234	4.2345

VI. CONCLUSION

We presented techniques to model a single session and models that can be used to examine the performance of a multi-tier web site. The JSQ infinite server model allows session imbalance to be determined and the approximate MVA can then be used to obtain range of performance measures. Designers and planners can use these to model different web site configurations and architectures. Current work is on developing an approximate method to determine joint probabilities in the JSQ infinite server model and to address weaknesses in the MVA algorithm for very high values of C^2 .

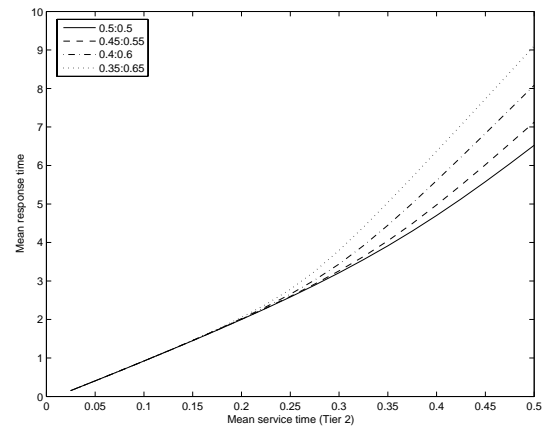


Figure 6. Mean response time with 100 sessions ($C^2=10$).

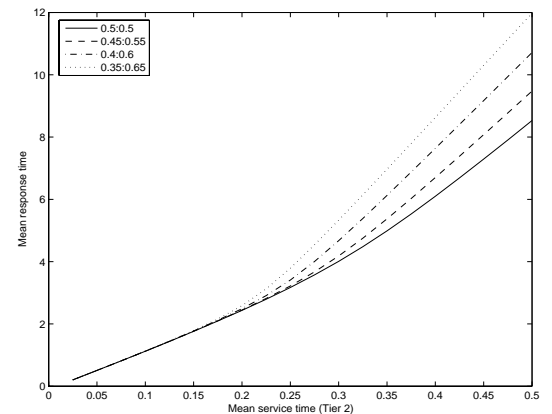


Figure 7. Mean response time with 121 sessions ($C^2=10$).

REFERENCES

- [1] Menasce, D. A. and Almeida, V. A. F., "Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning", 1st edition, Prentice Hall, 2000
- [2] Hariharan, R.; Kulkarni, V.G. and Stidham, S., "Optimal control of two infinite server queues." Proc. 29th IEEE CDC, Hawaii, 1990.
- [3] Yao, H. and Knessl, C., "On the infinite server shortest queue problem - symmetric case", Stochastic Models, 21(1): 101-132, 2005
- [4] Winston, W., "Optimality of the shortest line discipline". Journal of Applied probability, 14, 181-189, 1977.
- [5] Crovella M. E., and Bestavros, A., "Self-similarity in World Wide Web traffic: evidence and possible causes". IEEE/ACM Trans. Networking, 5(6):835-846, December 1997.
- [6] Arlitt M. F. and Williamson, C. L., "Web server workload characterization: the search for invariants". IEEE/ACM Trans. Networking, 5(5):631-645,1997
- [7] Takagi, H., "Queueing Analysis -Vol. 1: Vacation and Priority Systems", Part 1, North-Holland, 1991.
- [8] Transaction Processing Performance Council <http://www.tpc.org/>
- [9] Ugaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M. and Tantawi, A., "An analytical model for multi-tier internet services and its applications". Sigmetrics'05, June 6-10, 2005, Banff, Alberta, Canada.
- [10] Slothouber, L. P. "A Model of Web Server Performance.", <http://louvx.biap.com/white-papers/performance/overview.html>
- [11] Reiser, M., Lavenberg, S.S.: "Mean Value Analysis of Closed Multichain Queueing Networks", J. of the ACM, Vol. 27, No. 2, Apr. 1980
- [12] Schweitzer, P., "Approximate analysis of multiclass closed networks of queues". In Proc. Int. Conf. on Stochastic Control and Optimization, Amsterdam, 1979, 25-29.
- [13] Eager, D. L., Sorin, D. J. and Vernon, M. K., AMVA techniques for high service variability, ACM Sigmetrics 2000, California, 2000.