Priority Scheduling Service for E-Commerce Web Servers

Muhammad Younas[1], Irfan Awan[2], Kuo-Ming Chao[3], Jen-Yao Chung[4]

[1]*Department of Computing, Oxford Brookes University, Oxford, United Kingdom*

m.younas@brookes.ac.uk

[2]*Department of Computing, University of Bradford, Bradford, United Kingdom*

i.u.awan@brad.ac.uk

[3]*Department of Computer and Network Systems, Coventry University, Coventry, United Kingdom*

k.chao@coventry.ac.uk

[4]*IBM T.J Watson Research Center, Yorktown Heights, New York, USA*

jychung@us.ibm.com

Abstract: Service scheduling is one of the crucial issues in E-commerce environment. E-commerce web servers often get overloaded as they have to deal with a large number of customers' requests — for example, browse, search, and pay, in order to make purchases or to get product information from E-commerce web sites. In this paper, we propose a new approach in order to effectively handle high traffic load and to improve web server's performance. Our solution is to exploit networking techniques and to classify customers' requests into different classes such that some requests are prioritised over others. We contend that such classification is financially beneficial to E-commerce services as in these services some requests are more valuable than others. For instance, the processing of 'browse' request should get less priority than 'payment' request as the latter is considered to be more valuable to the service provider. Our approach analyses the arrival process of distinct requests and employs a priority scheduling service at the network nodes that gives preferential treatment to high priority requests. The proposed approach is tested through various experiments which show significant decrease in the response time of high priority requests. This also reduces the probability of dropping high priority requests by a web server and thus enabling service providers to generate more revenue.

Keywords: *Priority scheduling, Web server performance, E-commerce requests*

1. Introduction

E-commerce services are rapidly growing as more and more users and corporations make use of the Internet. They are becoming increasingly popular as they are easy to use, faster and cheaper to acquire. For example, buying flight tickets from the web are generally cheaper than high street travel agents. However, the side effect of the popularity of E-commerce services is the dramatic increase in the workload of E-commerce web servers. Such increase in the workload creates new challenges for managing the performance of web servers. Performance of web servers has therefore become one of the key factors to the successful operation of E-commerce services such as online shopping, auctions, and stock trading. Customers are not willing to use an E-commerce web server if its response time is slow. Thus businesses are forced to employ different strategies in order to improve the performance of their web servers. For instance, E*TRADE FINANCIAL (E*TRADE FINANCIAL 2007) provides 2 seconds order execution guarantees for processing stocks' requests.

Extensive research has been carried out in order to improve the performance of web servers. For instance, clustering of web servers, cache servers, web site optimization and scheduling mechanisms have been developed with the aim of improving web server's performance. Clusters of multiple web servers are deployed to avoid server overload and improve the response time (Menascé 2002b). Cache servers are also used to improve the performance of web servers (VanderMeer et al 2004). Website Optimization, LLC (Website Optimization 2007) is a web performance and Internet marketing firm which provides services for the optimization of web sites (such as optimizing HTML contents, web page graphics, etc). Scheduling mechanisms are devised to schedule requests such that performance can be improved (Elnikety et al 2004, Harchol-Balter et al 2003, McWherter et al 2004). Our work in this paper investigates the performance of web

servers by considering the scheduling of E-commerce requests. In addition, various benchmarks, for example, (Transaction Processing Performance Council 2007, Menascé 2005, Menascé 2002a) have been developed for the performance of E-commerce applications.

In order to acquire desired E-commerce services customers interact with E-commerce web server through a series of requests such as searching a web site for airfares or buying flight tickets. A typical example of user interaction with an E-commerce site is the online shopping (as in Amazon - www.amazon.com). In order to find a particular product user first either enters a keyword or clicks on a category link of a required product. User then may select a product to buy and move to add-to-cart operation or he may browse another product or quit the site. As shown in Figure 1, such user interaction with a web site can be modelled as a generalized state transition diagram (c.f. Menascé et al 1999). It represents different states S_1, S_2, …. S_6 and the probabilities P_1, P_2,….P_{11}. These states correspond to different requests such as browse, search, add-to-cart and payment requests. Transition from one state to another is represented with different probabilities. For example, P1 represents the probability that a user may move from state, S_1, to state, S_2 (e.g., moving from browse to search state). Note that this diagram is for illustrative purposes and it does not show all the state transitions and probabilities involved in a user interaction with an E-commerce site.

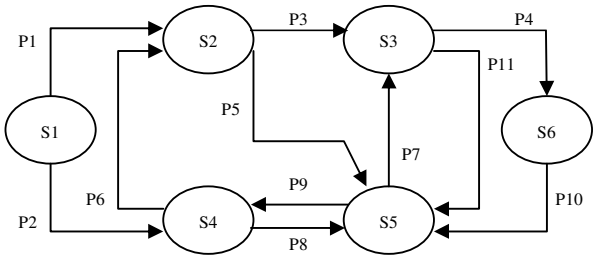

Figure 1. State Transition Diagram

Based on the current research studies, it is observed that the number of search and browse requests is significantly higher than add-to-cart and payment requests. According to Menascé et al (1999), the percentage of customers who buy items is significantly lower than buyers who usually use E-commerce service to find information such as air fares or book prices, without buying anything. Similarly, other research studies report that the number of customers (who buy items from the Internet) is 5% (see Nielsen (2007) for details). The large number of search and browse requests has performance consequences as they severely affect the processing and response time of high priority requests such as payment or add-to-cart. To alleviate such problems it is crucial to assign priorities to the high priority requests and provide differing levels of performance. When both high and low priority requests compete for web resources, high priority requests should complete more quickly on average than low-priority requests. For example, high priority requests such as payment requests should be prioritised over browsing requests. Generally, the processing of payment requests is more important to the service provider than a browsing request.

The above observations motivate our approach for the priority scheduling of requests. The proposed approach exploits the capabilities of the underlying networks on which E-commerce requests are carried out. It is based on our previous work (Younas et al 2006) and is developed using active network technology in order to insert customized programs into network nodes. Active networks have been used to improve the performance of distributed applications such as online auctions, mixing sensor data, and transaction processing (Legedza et al 1998, Younas and Awan 2003, Awan and Younas 2004).

Potential contributions of our work are as follows. It significantly reduces the queuing delay of high priority requests. Consequently, performance of such requests is improved as response time of the nodes responsible for making decision is reduced. Unlike existing solutions (Menascé et al 1999, Harchol-Balter et al 2003, Elnikety et al 2004), our approach does not demand pre-requisite information (e.g., registered users, server log file, or size of requested files) in assigning priorities

to E-commerce requests. The proposed approach assigns priorities based on the type of requests such as payment, add to cart and browse requests. Some services (e.g., opodo.co.uk) allow users to purchase products without registration. Thus requests should be prioritized according to their types. Further, our approach maintains the autonomy of web servers as they are not required to undergo modifications in order to accommodate the new scheduling service. Instead, such service is deployed at the active network nodes that perform the tasks of request scheduling. In addition, deploying the priority scheduling mechanism at the active network nodes relieves web servers from the extra processing incurred as a consequence of prioritizing E-commerce requests.

The remainder of the paper is organised as follows. Section 2 reviews related work and identifies the nature of the problem. Section 3 presents the proposed approach. Section 4 discusses experimental results. Section 5 concludes the paper.

2. Review of Current Solutions

This section reviews related work on the performance of web servers and related applications. Menascé et al (1999) present an interesting methodology for workload characterisation of E-commerce web sites. It also proposes a Customer Behaviour Model Graph (CBMG) that describes the behaviour of customers who follow similar navigational patterns in submitting requests to E-commerce web sites. In these sites online shoppers issue requests such as browse, search, and pay. CBMG is used to describe the sequence of such requests. CMBGs are constructed by analysing logs of an E-commerce site that contains information related to user's profile based on the previous navigation patterns. Different users are characterised by different CBMGs. For example, one CBMG can be constructed for occasional buyers who usually use E-commerce site to find information such as air fares, itineraries, and books prices, without buying anything. Another CBMG can be constructed for users who have high probability of buying products from the E-commerce site.

CBMG is useful to determine the behaviour of customers visiting E-commerce site. For example, when a customer starts navigating web site, the web server can use the profile information (stored in the log file) and assigns different priorities based on a user profile. However, it incurs processing overhead in constructing the CBMG using the past information stored in the log files that describes the customers profiles — CBMG is constructed even if a customer visiting a web site does not buy items. Another alternative is to use registration information to classify customers into occasional buyers or heavy buyers. That is, registered users are more likely to buy (heavy buyers) as compared to non-registered users who are less likely to buy (occasional buyers). However, it is unrealistic to assume that registered users will buy items each time they visit an E-commerce web site.

The work presented in (He and Yang 2000) studies the performance of web servers by taking into account static web pages as well as dynamic web pages generated through CGI, servlets and database queries. This study measures system throughput and user response times in five different architectural scenarios. It reveals that the performance behaviours of web servers serving dynamic contents are different from the ones serving static contents. This study gives some interesting observations but it does not consider network related performance issues.

Elnikety et al (2004) implements a proxy server, called Gatekeeper that enables admission control (overload control) and provides differentiated scheduling of requests to improve response time. Admission control is based on the principle that a maximum load should be kept just below the capacity of an E-commerce system. This prevents system overload and also achieves high throughput. Differentiated scheduling of requests is based on the shortest-job-first (SJF) policy which assigns higher priority to the shorter jobs. It also proposes an aging mechanism in order to prevent starvation of longer jobs. This mechanism enforces an upper bound on the amount of time a request is delayed in the queue. Gatekeeper is claimed to achieve improved performance and maintains stable behaviour during work load. However, there are issues with SJF policy. First, SJF policy cannot improve performance if all the requests are homogeneous requiring same service time. Second, it requires that the size of requests or the target data is to be known in advance. Third, most of the requests are of same size in terms of processing time. Thus SJF may not be an effective policy.

Harchol-Balter et al (2003) employ a pre-emptive version of SJF scheduling, called SRPT (Shortest Remaining Processing Time) first policy. SRTP is used to improve the performance of web servers. However, this work considers static web pages such that priority is given to requests for small files or requests with shortest remaining file size. McWherter et al (2004) propose priority mechanism for transactions in classical database systems. This work presents a detailed

analysis of the resource utilisation by transactions in a database system. It also improves the performance of high priority transactions in classical database system.

Singhmar et al (2004) propose a LIFO-Pri priority scheduling scheme in order to give service priority to revenue generating (such as payment) requests over the browsing requests. This scheme is based on a large number of queues which are extremely difficult to manage. The proposed scheme works by moving revenue generating requests from one queue to another queue based on its current state during its processing. This needs keeping track of requests throughout their entire execution. It may be manageable for fewer requests but will show performance degradation for larger numbers of requests. Our previous work (Awan and Younas 2004, Younas and Awan 2003) employs active network priority scheduling mechanisms in order to improve the performance of transaction commit protocols in Web-database applications. These approaches gives preferential treatment to the processing of decision messages (such as transaction commit, abort, compensate) as compared to data related messages.

3. The Priority Scheduling Service for E-commerce

This section presents our proposed priority scheduling service which is implemented at the network nodes in order to assign different priorities to E-commerce requests.

As described above users interact with web servers through a series of requests in order to acquire required E-commerce services. For example, to find a particular product user first either enters a keyword or clicks on a category link of a required product. User's request is sent to the web server which in turns passes the request to the application server and then to the database server. As shown in Figure 2, these services are generally implemented in an architecture that involves client systems, network, web servers, application servers and data servers. Web servers typically serve static contents such as HTML pages or still images. Application servers (e.g., BEA WebLogic, IBM WebSphere) are commonly used to generate dynamic web contents by running scripts written in a number of languages such as Active Server Pages (ASP), Java Server Pages (JSP), and Perl. Scripts execute the necessary logic to process customers' requests by contacting various resources in order to retrieve, process, and format the requested content into customer deliverable web pages. Our work is not concerned with the performance aspects of the application server or the database server (VanderMeer et al (2004) gives details on the performance evaluation of the application and database servers). It focuses on web server performance and how it can be improved by exploiting the capabilities of underlying networks.

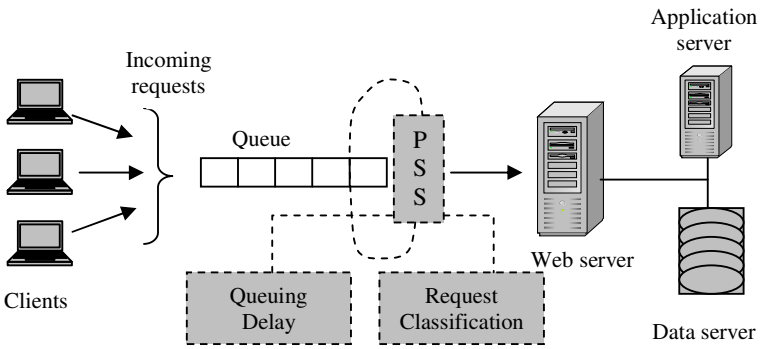

Figure 2. Architecture of the Proposed System

The proposed approach, called *priority scheduling service* (PSS), is based on the implementation of the active network priority scheduling mechanism at network nodes (see Figure 2). PSS assigns different priorities to E-commerce requests — high priority requests such as

payment and add-to-cart get higher priorities, while other requests such as browse and search are assigned lower priorities.

Unlike existing solutions (Menascé et al 1999, Harchol-Balter et al 2003, Elnikety et al 2004), PSS does not demand pre-requisite information (e.g., registered users, server log file, or size of requested files) in assigning priorities to E-commerce requests. PSS assigns priorities based on the type of requests. The objective of this work is to reduce network queuing delay involved in the message communication of E-commerce requests. Such reduction of queuing delay significantly improves the performance of high priority E-commerce requests. In order to reduce queuing delays we take into account the priority scheduling mechanism of active networks. Fundamental principle of these mechanisms is the provision of preferential treatment to some requests as compared to others. One of the useful priority scheduling mechanisms is the pre-emptive resume (PR) scheduling (Awan and Kouvatsos 2002, Awan and Kouvatsos 1999). In the proposed approach, PR mechanism is employed at each network node involved in the processing of E-commerce requests. According to PR, the arriving high priority message pre-empts the low priority message being processed. The pre-empted message resumes its processing soon after the high priority message is processed. In PR mechanism, each node in the network is equipped with a finite capacity buffer that stores the incoming messages. The total time that a message spends in the node is the sum of the waiting time and the processing time. Waiting time for each message is the sum of processing times for all the messages in front of it. The PR scheduling algorithm is described in Figure 3. The required parameters of PR scheduling are given in Table 1.

Table 1: Priority Scheduling Service Parameters

Inputs	Description
N	Buffer capacity
R	Number of classes for incoming requests
Arrival parameters	
λ_i	Arrival rate for $i=1,2,\ldots,R$ classes of requests
C_{ai}^2	SCV – to represent traffic burstiness of arriving requests
Service parameters	
μ_i	Service rate for $i=1,2,\ldots,R$ classes of requests
C_{si}^2	SCV – to represent burst departure of processed requests
Notations	
n_H	Number of high priority requests in the buffer
n_L	Number of low priority requests in the buffer
H	High priority
L	Low priority

Figure 3. Algorithm: PR Priority Scheduling

```
IF (Priority(arriving request) = H) & (n_H + n_L < N) THEN
    IF (n_H =0) & (n_L =0) THEN
        Process this request

    ELSE IF (n_H =0) & (n_L > 0) THEN
       {
            - pre-empt the request in service
            - start high priority request
       }
    ELSE
      queue arriving request behind high priority
      requests
    End-IF
End-IF

IF (Priority(arriving request) = L) & (n_H + n_L < N) THEN
```

```
      IF (n_H =0) & (n_L =0) THEN
          Process this request
      ELSE IF (n_H ≥ 0) & (n_L ≥ 0) THEN
          queue arriving request behind all requests.
      ELSE IF (n_H + n_L = N) THEN
          drop every arriving request.
      End-IF
End-IF
```

Outputs:

- Mean response time: total time spent in the system
- Throughput: Total number of completed requests
- Dropping probability: number of packets found the queue full upon their arrival over the total number of arrivals

Employment of PR reduces the queuing delays at the network nodes involved in the processing of E-commerce requests. In order to calculate the queuing delay each network node is modelled as a queuing system with finite capacity. The arriving external traffic at each node is bursty as requests from various E-commerce applications can arrive simultaneously. This has been modelled using Compound Poisson Process (CPP). Each node may have multiple processors and hence can execute various requests simultaneously. This concurrent execution has been modelled using a Generalised Exponential (GE) distribution. Based on such information, each node has been analysed as a GE/GE/1/N queuing system with PR scheduling discipline to give preferential treatment to arriving messages. This analytical solution provides closed form expression to calculate the queuing delay at each network node as described below.

The Network Delay: Network queuing delay of the E-commerce requests is considered as an important element in the performance analysis of E-commerce requests given the heavy traffic of the Internet, for example, traffic generated by E-commerce requests, news channels, video streaming, chat rooms and so on.

In order to calculate queuing delay, we consider a stable single server GE/GE/1/N queue under a priority PR scheduling discipline. R (>1) represents multiple classes (low and high priority) of requests. For each class, i (i=1,2,…,R), let λ_i be the mean arrival rate, C_{ai}^2 be the inter-arrival time squared coefficient of variation (SCV), μ_i be the mean service rate and C_{si}^2 be the service time SCV. Let at any given time, n_i ($0 \le n_i \le N$), $\sum_{i=1}^{R} n_i \le N$, be the number of class i requests in the queue (waiting and/or receiving service), $S=(n_1,n_2,\ldots,n_R)$ be a joint queue state and T be the set of all feasible states S. The form of the state probability distribution $P(S)$, $\{S \in T\}$ of a GE/GE/1/N/PR priority queue, can be characterized by maximizing the entropy functional,

$$H(P) = -\sum_{S \in T} P(S) \log P(S) \quad \text{.....(1)}$$

This is subject to prior information expressed in terms of the normalization and, for each class i (i=1,2,…,R), the marginal constraints of server utilization, U_i ($0<U_i< 1$), busy server probability θ_i ($0 < \theta_i < 1$) with $n_i>0$, mean queue length, L_i ($U_i \le L_i< N$) and conditional full buffer state probability, given that a class i request is in service, ϕ_i ($0 < \phi_i < 1$), satisfying the flow balance equations, namely

$$\lambda_i (1-\pi_i) = \mu_i U_i, i = 1,2,\ldots,R,$$

where π_i is the marginal blocking probability that an arriving request of class i finds N messages in the queue. By employing Lagrange's method of undetermined multipliers and after some manipulation, the probability distribution of requests can be expressed by

$$P(S) = \frac{1}{Z} g_i \xi_i x_i^{n_i} y_i^{f_i(S)} \left(\prod_{j=i+1}^{R} x_j^{n_j} \xi_j^{h_j(S)} \right), i = 1,...,R, \quad (2)$$

where Z is the normalizing constant, $\{g_i,\xi_i,x_i, y_i\}$ are the Lagrangian coefficients corresponding to constraints $\{U_i, \theta_i, L_i,\phi_i\}$, respectively and $\{h_i(S),f_i(S)\}$ are suitably defined auxiliary functions (Awan and Kouvatsos 2002). Utilizing this product-form solution, the closed-form expressions for basic performance metrics such as mean marginal and aggregate delays, Q_i and Q, respectively, can be obtained (c.f., (Awan and Kouvatsos 2002)). In particular, the mean delays can be clearly determined (via Little's Law) by

$$Q_i = \frac{L_i}{\hat{\lambda}_i} \quad (3)$$

where $\hat{\lambda} = (1 - \pi_i)$ is the mean effective arrival rate of class i requests and

$$Q = \sum_{i=1}^{R} \frac{\hat{\lambda}_i}{\hat{\lambda}} Q_i, \qquad \hat{\lambda} = \sum_{i=1}^{R} \hat{\lambda}_i. \quad (4)$$

4. Evaluation

This section first illustrates the simulation model developed to test the performance of the proposed approach. It then discusses the experimental results.

4.1 Simulation Model

In order to evaluate the performance of the proposed approach, we develop a simulation model based on the requests (such as search, add to cart, payment) generated. The simulation model is developed using Queuing Networks Analysis Package (QNAP2) (Badel et al 1981) which provides analytical and simulation techniques for performance analysis. Experimental study has been carried out using the analytical model presented in Section 3. This model takes into account different input parameters to see their impact on performance measures of mean response time. The input parameters include:

- *traffic load:* rate at which requests are arriving to the system
- *mean batch size:* simultaneous arrival of these requests
- *server capacity:* the rate at which server is processing the arriving requests
- *buffer size:* the total accommodation available at the input port of the server to temporarily store the arriving requests when the server is busy

The mean response time represents the total system time for processing a request from its arrival till its completion. This includes the time spent in the buffer waiting for the server and the time taken by the server for its processing. Note that our experiments are based on estimated values of the above parameters as they vary according to network traffic and server load.

4.2 Experiments

The experiments cover a wide range of input parameterisations and demonstrate particularly the mean response time for various types of E-commerce requests by assigning different priorities. These experiments also take into account the burstiness property of the requests to represent simultaneous arrival of requests from a large number of users. We conduct the following experiments:

4.2.1 Response Time: Two Classes of Requests

In this experiment we consider two types of requests; high priority (class 1) and low priority (class 2) requests. Figure 4 shows the results of the experiments that calculate the mean response time of high and low priority requests. It takes into account different number of requests (traffic load). This experiment shows that increasing the traffic load will slightly affect the mean response time for high priority requests. However, the mean response time increases very rapidly for the low priority requests. It is shown that the proposed approach significantly improves the efficiency of the high priority requests.

Figure 5 shows the system throughput for two classes of requests under the same priority scheduling discipline, PR. By increasing the traffic load, the throughput for high priority requests will rapidly increase as compared to low priority requests. This increase is due to the pre-emption of low priority request from the service whenever there is an arrival of high priority requests.

Figure 6 presents the effect of increasing traffic intensity for both classes of requests on the packet drop probabilities. It can be seen that the packet dropping probabilities for both classes of requests steadily increase for increasing traffic. This is obviously due to the finite capacity of buffer for incoming requests. Another interesting information presented in this figure is that there is no significant difference between the dropping probabilities for the two classes of requests. The main reason for this similar behaviour for two classes of requests is that the proposed PR scheduling scheme only deals with the service priority and never drops packets of low priority requests from the buffer once they get any place. Thus, for the same external traffic load, PR will always give close dropping probabilities when there is no space priority.

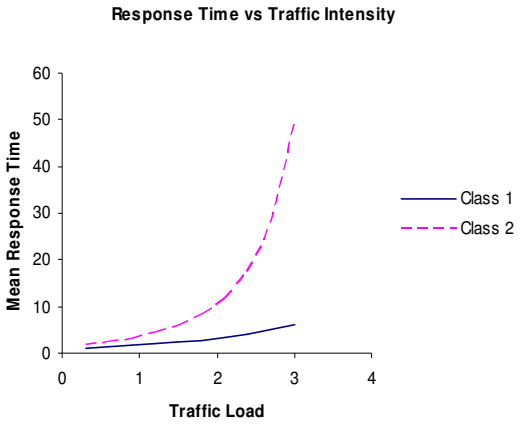

Figure 4. Response Time for Two Types of Requests

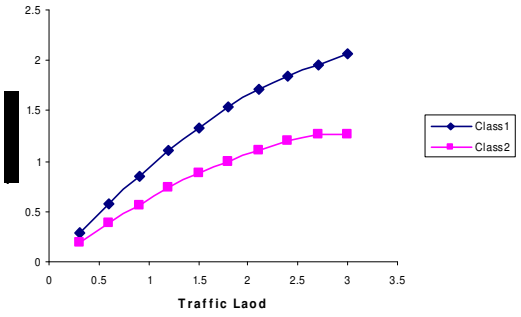

Figure 5. Comparison of System Throughput for Two Classes of Requests under PR Discipline

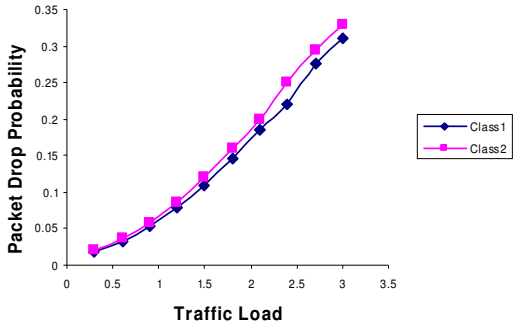

Figure 6. Comparison of Packet Drop Probability for Two Classes of Requests under PR Discipline

4.2.2 Response Time: Three Classes of Requests

In this experiment we consider three types of requests. The aim is to show that our approach is capable of assigning more than two types of priorities to E-commerce requests. This experiment further classifies high priority requests into two classes. For example, it assigns higher priority to the payment requests than add-to-cart requests. That is, if a payment request arrives at the node then it will get higher priority than add-to-cart request. But the low priority requests (such as search and browse) still get lower priority than the above classes of high priority requests.

Figure 7 shows the results of this experiment. Class-1-A shows the most high priority requests. Class-1-B shows the second high priority requests. Class-2 shows the low priority requests. Similar to the above experiment, we take into account different number of requests (with varying traffic load).

The experiment shows that Class-1-A requests have a lower mean response time than Class-1-B and Class-2. Class-1-B has a lower mean response time than Class-2.

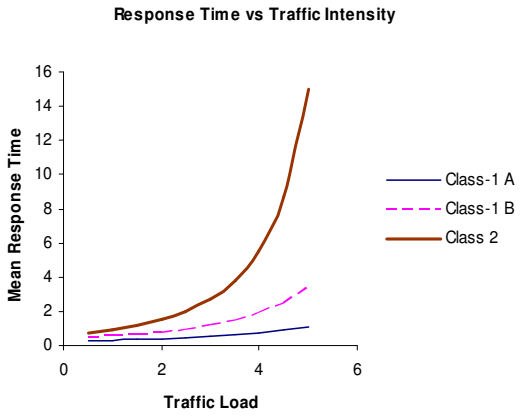

Figure 7. Response Time for Three Types of Requests

4.2.3 Mean Response Time

In this experiment we have compared the mean response times for two classes of traffic under PR scheduling discipline with those under FCFS discipline. Figure 8 shows that processing time of high priority requests under PR discipline is lower than processing these requests under FCFS

discipline. Figure 9 shows that the performance of low priority request deteriorates under PR as compared to FCFS discipline. This is mainly because high priority requests keep server busy during their presence in the system while low priority requests wait in the queue.

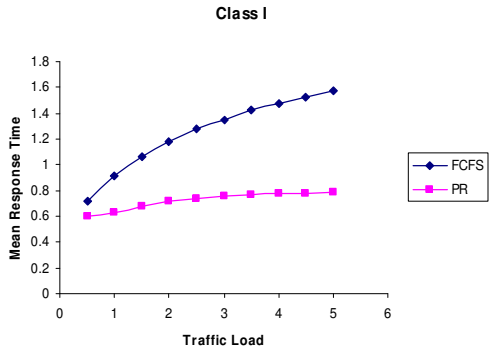

Figure 8. Comparison of Mean Response times for Class 1 Requests under FCFS and PR Discipline

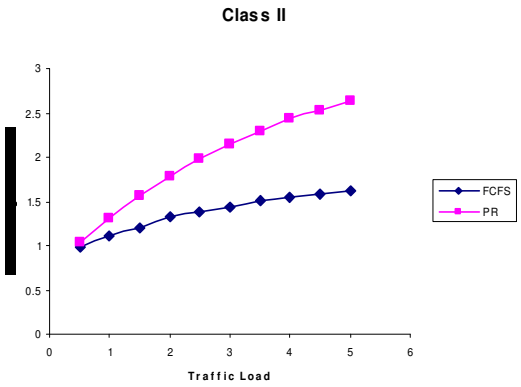

Figure 9. Comparison of Mean Response Times for Class 2 Requests under FCFS and PR Discipline

5. Conclusion

This work investigated the performance of E-commerce requests which is one of the key issues in E-commerce research. In it, we presented a network-centric approach in order to achieve improved performance in the processing of the high priority E-commerce requests. Current approaches incorporate different strategies in improving the performance of E-commerce requests. However, they fall short of considering the capabilities of the underlying networks which can affectively be used to improve the performance of E-commerce requests. Our approach exploits the capabilities of the underlying networks using effective priority scheduling mechanism that treats different requests differently. We conducted different experiments in order to evaluate the proposed approach. These experiments demonstrated significance performance improvement of high priority requests.

References

Awan I, Kouvatsos D (1999) Approximate Analysis of Arbitrary QNMs with Space and Service Priorities, Performance Analysis of ATM Networks, Kluwer Publishers, pp. 497-521

Awan I, and Younas M (2004) Analytical Modelling of Priority Commit Protocol for Reliable Web Applications, Proceedings of the 19th ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus

Awan I, Kouvatsos D (2002) Approximate Analysis of Arbitrary QNMs with HoL Priorities, CBS Buffer Management Scheme and RS-RD Blocking, Proceeding of 18th UKPEW, Glasgow, UK, pp. 15-26

Badel M, Chandresris D, Guillemaud J-J, Potier D, Saintoyant P-Y, Veran M (1981) QNAP2 Reference Manual, Cii Honeywell Bull and INRIA

Elnikety S, Nahum E, Tracey J, and Zwaenepoel W (2004) A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites, Proc. of ACM WWW Conference, New York, USA

E*TRADE FINANCIAL (accessed on 20 February 2007), https://us.etrade.com/e/t/home

Website Optimization (2007), LLC (accessed on 20 February 2007), http://www.websiteoptimization.com/

Harchol-Balter M, Schroeder B, Bansal N, Agrawal M (2003) Size-Based Scheduling to Improve Web Performance, ACM Transactions on Computer Systems, 21(2), pp. 207–233

He X, Yang Q (2000) Performance Evaluation of Distributed Web Server Architectures under E-Commerce Workloads, Proceedings of the 1st International Conference on Internet Computing (IC'2000), Nevada, USA, pp. 285-292

Legedza U, Wetherall D, Guttag H (1998) Improving the Performance of Distributed Applications Using Active Networks, in Proceedings of the 17th Conference on Computer Communications (INFOCOM), San Francisco, California, April 1998, IEEE, pp. 590-599

McWherter D, Schroeder B, Ailamaki A, Harchol-Balter M (2004) Priority Mechanisms for OLTP and Transactional Web Applications, 20th International Conference on Data Engineering (ICDE 2004), Boston, USA

Menascé DA, Almeida VAF, Fonseca R, Mendes MA (1999) A Methodology for Workload Characterization of E-commerce Sites, ACM Conference on Electronic commerce, Denver, Colorado, USA, pp. 119-128

Menascé DA (2005) MoM vs. RPC: Communication Models for Distributed Applications, IEEE Internet Computing, Vol. 9, No. 2, ISSN: 1089-7801, pp. 90-93

Menascé DA (2002a) Trade-offs in Designing Web Clusters, IEEE Internet Computing, Vol. 6, No. 5, ISSN: 1089-7801, pp. 76-80

Menascé DA (2002b) TPC-W: A Benchmark for E-commerce, IEEE Internet Computing, Vol. 6, No. 3, ISSN: 1089-7801, pp. 83-87

Nielsen J (2007) Why People Shop on the Web
http://www.useit.com/alertbox/990207.html

Singhmar N, Mathur V, Apte V, Manjunath D (2004) A Combined LIFO-Priority Scheme for Overload Control of E-commerce Web Servers, International Infrastructure Survivability Workshop (IISW'04) Overloads, Attacks and Failures: the Trade-off against Time in conjunction with the 25th IEEE International Real-Time Systems Symposium (RTSS04), Portugal

Transaction Processing Performance Council (2007), TPC-W: Transactional Web E-Commerce Benchmark
http://www.tpc.org/tpcw/

VanderMeer D, Datta A, Dutta K, Thomas H, and Ramamritham K, (2004) Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation, ACM Transactions on Database Systems (TODS), 29(2), pp. 403-443

Younas M, Awan I (2003) Efficient Commit Processing of Web Transactions Using Priority Scheduling Mechanism, 4th International Conference on Web Information Systems Engineering (WISE 2003), IEEE CS, Rome, Italy

Younas M, Awan I, Chao K-M (2006) Efficient Scheduling of Vital E-Commerce Requests, 2006 IEEE International Conference on e-Business Engineering (ICEBE 2006), Shanghai, China. pp. 496-503