Providing Differentiated Service from an Internet Server

Xiangping Chen and Prasant Mohapatra
Dept. of Computer Science and Engineering
Michigan State University, East Lansing, MI 48824

Abstract

Differentiated service has been proposed as a potential solution for bandwidth allocation and expected to be supported in the next generation Internet. However, a service differentiating Internet with best-effort servers may not meet the overall goals of the differentiated service. In this paper, approaches and performance issues on providing differentiated services from an Internet server are studied. Experimental study and analyses prove that under near-saturation of server utilization, differentiating service provides significantly better performance to high priority tasks compared to a traditional service mode. Quantitative performance estimation of different priority levels of tasks is presented. It is also observed that an enhanced shortest queue first task assignment scheme helps in decreasing the average response time of the server system.

I. INTRODUCTION

The Internet and the World Wide Web (WWW) provide a global publishing and information access infrastructure at low costs. Thousands of companies set up web sites for marketing and interactions with customers. Although some companies are highly successful in using the Web as their market routes, most companies are still reluctant to use it for core business transactions due to the problem of unguaranteed services in contemporary WWW servers.

Using high performance servers and broader network bandwidth is one solution to provide better Internet services, but the "bursty" nature of the Internet traffic and the rapid growth in the Internet user community makes it not only hard but also inefficient to scale up network bandwidth and server power to provide predictable Quality of Service (QoS).

Differentiated service, known as *diffserv* in the IETF community, has been proposed as an efficient and scalable solution to provide better service for next generation Internet (NGI) communication [1], which creates an "express-way" to some crucial flows through prioritized transmission. We believe that both the network as well as Internet servers must support differentiated services. Our objective is to design a prioritized Internet server system which can provide fast response to high priority tasks, minimize the performance penalty on low priority tasks without degrading the overall system throughput.

The remainder of this paper is organized as follows. Section 2 presents the web server model, admission control and scheduling issues. The simulation environment and results are discussed in Sections 3 and 4 respectively. Section 5 provides analysis of the results, followed by the concluding remarks in Section 6.

II. DISTRIBUTED SERVER MODEL

Figure 1 shows a distributed server system with centralized task routing. A server consists of four logical components, an initiator S_I, a scheduler Q, N task servers $S_i (i = 1...N)$, and a communication channel N_S. Incoming requests are queued awaiting admission from the initiator. Each accepted request is assigned a task, and each task is granted an appropriate priority level by the scheduler. The scheduler assigns tasks to each task server, and those task servers schedule and process tasks according to their priorities. Responses are sent back to clients through the communication channel. The capacity of the communication channel is determined by the Internet access point of both the clients and the server, and the backbone bandwidth.

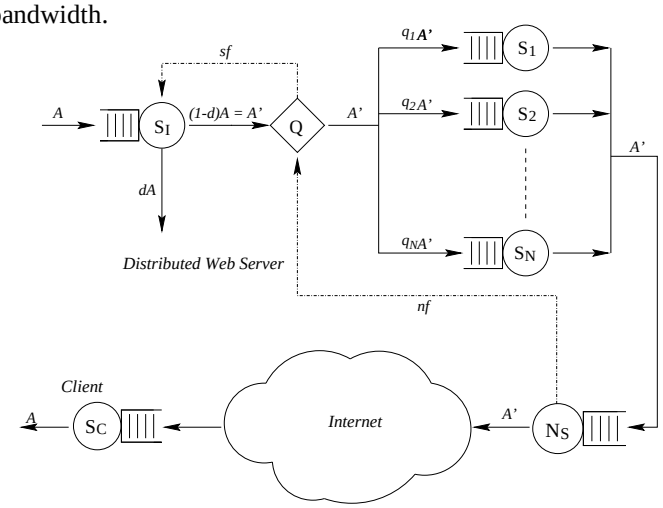

Figure 1: Queuing Network Model for Distributed Web Servers.

In a general queuing network, the response time of a task is the sum of response times from each component. Most components are sensitive to system overload [2]. If the request access rate and system resource consumption exceed certain thresholds, the system becomes unstable and response time for each task increases drastically. QoS can be assured by maintaining the system load within the thresholds, which can be achieved by admission control, scheduling, and efficient task assignment schemes (if multiple servers are available).

Admission control and scheduling is implemented in the initiator and the scheduler. Incoming requests are rejected by the initiator when access rate exceeding the system capacity. *sf* provides feedback of system load to the initiator. The scheduler schedules different priority level tasks to assure the system works in acceptable load condition. *nf* provides feedback of network load to the scheduler. In case of network overload, low priority multimedia tasks are discarded. Available bandwidth is used as a metric of channel load.

III. Simulation

We have implemented an event driven simulator for the experimental study of the model proposed in Section 2. In this study, we propose to generate workload from real trace files. Traces of ClarkNet [3] are chosen as raw data set, whose distribution is shown in Table 1.

Table 1
ClarkNet Data Distribution.

Item	HTM	IMG	AUD	VDO	DYN	OTH
Req. Rat. (%)	19.9	78.0	0.2	0.007	1.2	0.693
Acc. Rat. (%)	15.0	76.6	2.4	2.4	0.8	2.80
Tran. Sz (KB)	7.43	9.67	135.1	3,514.8	6.63	37.12
Tran. CoV	2.14	1.66	1.24	0.35	3.31	3.89

The "raw" traces are overlapped on a time base to generate different level of workload and preserve "bursty" nature of request flow. The traces of ClarkNet contains 3,328,587 requests in two weeks period. Timestamps have 1 second resolution, and logs with same timestamps are assumed to distributed uniformly in a 1 second time period. There are two reasons for choosing ClarkNet traces. First, it represents workload for a typical commercial Internet content provider. Second, the web documents access distribution are still popular in current Internet servers, which is indicated in [4].

For each web task, service time is the time spent in URL parsing, user authentication, file location and transmission, and logging. The parameters for task processing were obtained from [5]. For a cache hit or a CGI request, the service time is dominated by CPU computation time; and for a large sized file request, it is dominated by I/O processing time.

The effectiveness of a scheduling scheme is measured in terms of mean response time and mean slowdown [6]. Mean response time is defined as the time between the acceptance of the request and the completion of the service. It indicates the actual time needed to finish a task. On the other hand, task slowdown is a metric of user tolerance. A user is often willing to wait longer time for a big task. The slowdown of a task is the ratio of its response time to its service time.

IV. Results

A. Effectiveness of Priority Based Scheduling

In the Internet environment, both access interval and service time distribution are significantly different from widely used synthetic workloads. Therefore, we have used real traces for performance evaluation. The high variance of the inter-arrival time and service rate deteriorates system performance. As we can see from Figures 2 and 3, with the increase in server utilization, response time increases much faster under high utilization. Curve (c) in Figure 2 is the slowdown curve of requests of our model without priority distinction. Curve (d) is the slowdown curve of a general M/M/1 queuing system. The disparity between curve (c) and (d) indicates the impact of self-similar traffic on the system utilization. The utilization of 0.5 in the simulator leads to about the same level of delays as utilization of 0.8 in a Poisson distribution.

Performance degradation of high priority tasks with priority-based scheduling happens at a much higher utilization compared to the non-priority-based model. Curves (a) and (b) in Figure 2 are the slowdown curves of low priority requests and high priority requests, respectively. The ratio of high priority is 0.5 and is uniformly distributed in the whole arrival sequence. The introduction of priority queuing causes a steep rise of low priority requests response time at a system utilization of 0.4. High priority requests incur low delay even when the system approaches full utilization.

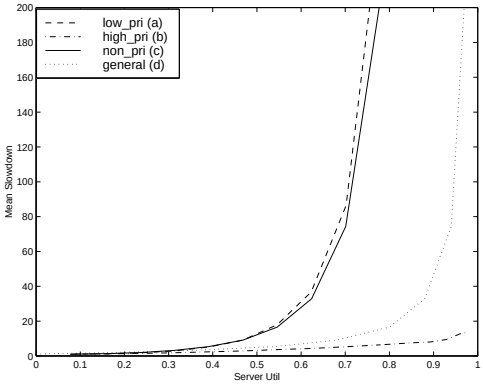

Figure 2: Task slowdown.

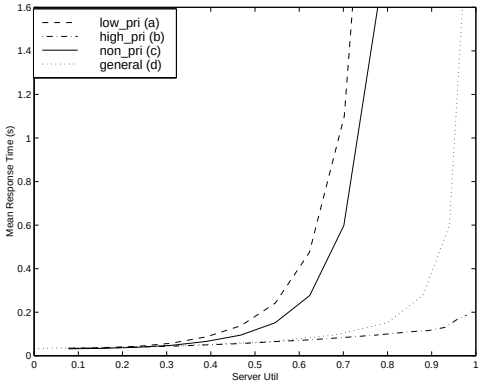

Figure 3: Task response time.

B. Maximum High Priority Ratio

Our next experiment explores ratio, the relationship between the increase in high priority ratio and the region of the "knee of the curves" in the system. Figures 4 and 5 show the delay curves of high priority tasks with high priority ratio varying from 0.5 to 0.9. With the increase in high priority ratio, the curve gets closer to the original non-prioritized system curve, and the margin of benefit obtained from differentiating service diminishes.

Figure 5 shows the mean response time of high priority tasks with 95% confidence interval. The "performance knee" of the mean response time of high priority tasks are primarily determined by effective utilization of high priority tasks. The relationship between the whole system utilization and the performance degradation point of high priority tasks is not obvious.

Figures 6 and 7 show the mean slowdown and response time curves of low priority tasks with the high priority ratio ranging

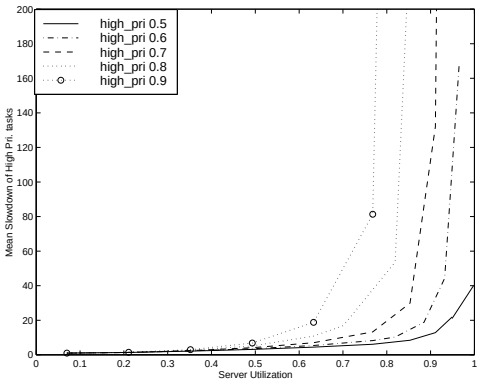

Figure 4: High priority ratio.

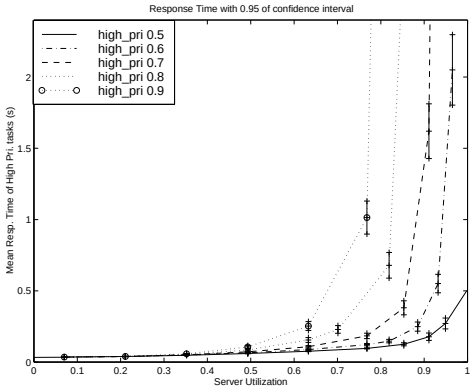

Figure 5: High priority ratio.

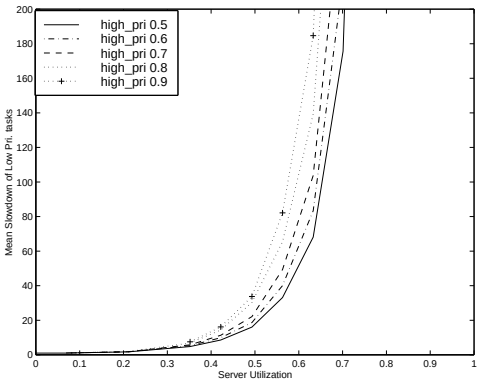

Figure 6: Low priority task slowdown.

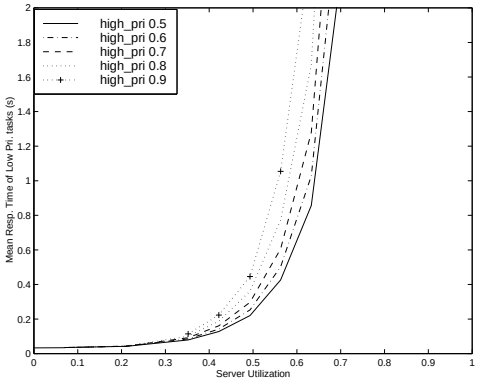

Figure 7: Low priority task resp. time.

from 0.5 to 0.9. With the increase in the high priority ratio, the system utilization that causes the steep rise of response time decreases. On the other hand, the spectrum of the occurrence of the "performance knee" is relatively narrow for low priority tasks, which reflects the decreased impact of low priority tasks with low percentage of access rate.

C. Task Assignment Schemes

In a distributed server environment, an appropriate task assignment scheme decreases the waiting time variance and thereby improves the system performance on the whole. We compare the three types of task assignment schemes experimentally. The first one is based on load balancing techniques, such as Round_Robin or Shortest_Queue_First schemes. The second type is resource reservation based, which decreases waiting time of a high priority task. The third one is preemption based, i.e., higher priority tasks can preempt lower priority tasks.

In load balancing type of task assignment schemes, Round_Robin (RR) assigns tasks in rotation order of servers without considering the priority of tasks. Shortest_Queue_First (SQF) is proven to be optimal in previous studies, and assigns a new task to the server with least number of waiting tasks. To adapt to the differentiated service environment, an Enhanced_SQF (E_SQF) scheme is introduced in which a new task is assigned to the server with least number of waiting tasks with equal or higher priority tasks than the new task. Figures 8 and 9 compares the performance of the above three schemes. Compared to the RR scheduling, SQF helps to decrease the variance of waiting queue length, thus the response time. E_SQF outperforms SQF when the high priority access rate is acceptable, and decreases the mean response time of high priority tasks, but there is no significant difference in these two schemes under high load.

In a non-preemptive environment, preserving some resources for high priority tasks decreases the chance of high priority tasks waiting for lower priority tasks in service. With the increase in reserved resources, the performance of the scheme by preserving resources gets close to the preemptive priority scheduling.

V. ANALYSIS

In this section, we try to derive a guideline for performance of high priority requests in such systems. Some notations used in the study are listed in Table 2.

A task's waiting time is decomposed into three parts: delay it encounters due to the task being in serviced upon its arrival; delay it experiences due to tasks enqueued upon its arrival; and delay due to higher priority tasks arriving after its arrival. For high priority tasks, the third part of waiting time does not exist.

$$W_h = W_2 + W_1 \tag{1}$$

In a non-preemptive system, from the viewpoint of a newly arrived task, the first part of the delay is due to the task found in service upon arrival. This delay is equal to the other tasks'

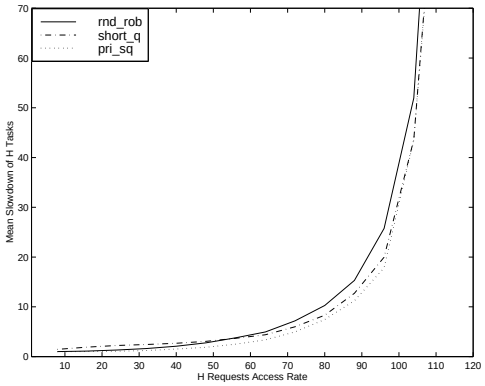

Figure 8: High priority task slowdown.

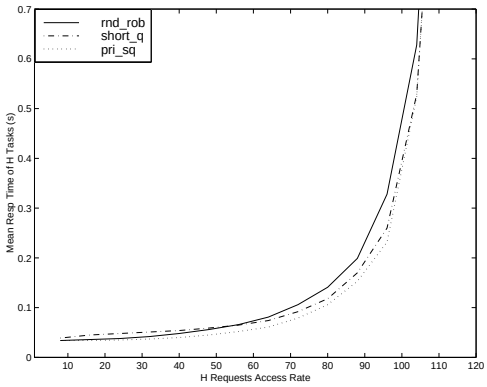

Figure 9: High priority task response time.

Table 2
Notations used in the study.

W_h	Mean waiting time for tasks in high priority group.
W_1	Residual life of a task in service.
W_2	Sum of execution time of queued tasks.
P_h	Probability of high priority tasks.
A_h	Arrival interval of high priority tasks.
X	Mean service time for tasks. $X = X_h = X_l$
T_h	Mean system time for tasks in high priority group.
N_{qh}	The number of queued high priority tasks.
P_l	Probability of low priority tasks in service.
W_l	Mean waiting time for tasks in low priority group.
T_l	Mean system time for low priority tasks.

residual life, and determined by the whole system utilization, see Equation 3. In the worst case, when each task enters the system, there is always a task in service, and the task in service is just beginning to execute. The average delay of the worst case is equal to mean service time.

$$W_1 = P_h * X_h + P_l * X_l \tag{2}$$

In the worst case, when each task enters the system, there is always a task in service, and the task it finds in service just begins its processing. Mean waiting time to the residual life of a service equals its service time, i.e., $W_1 <= X$

The second part is the execution time of queued tasks with equal or higher priority when a task enters the system. High priority tasks need not care about the waiting lower priority tasks. The time for waiting for the queued tasks only depends on the number of high priority tasks in the system.

$$\begin{aligned} W_2 &= X * N_{qh} && (3)\\ N_{qh} &= A_h * W_h && (4)\\ W_h &= W_2 + W_1 \\ &= X * A_h * W_h + W_1 \\ &= \frac{W_1}{1 - X * A_h} \\ &= \frac{P_h * X + P_l * X}{1 - X * A_h} && (5)\\ &\leq \frac{X}{1 - X * A_h} && (6) \end{aligned}$$

The upper bound of W_1 is X. The "shape of the curve" is mainly decided by A_h. To obtain a stable system response and throughput states, the high priority system utilization, $A_h * X$, shall not exceed the "knee" of the whole system utilization curve. The whole system utilization determines the P_h and P_l. When the arrival rate of high priority requests is bounded, reserving system resources decreases P_l, thus W_1. In a preemptive scheduling case, P_l equals to zero.

VI. Conclusion

The next generation Internet will demand differentiated services from web servers, which can be achieved through priority-based service. Servers need to provide high quality of service to high priority tasks even under high system utilization. In this study, we prove that under near-saturation of web server utilization, differentiated services provide significantly better services to high priority tasks compared to a traditional web server. We also present quantitative performance estimation of different levels of tasks.

VII. References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z.Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475. December 1998.

[2] L. P. Slothouber, "A model of web server performance," in *the Fifth International World Wide Web Conference*, (Paris, France), May 1996.

[3] http://ita.ee.lbl.gov/html/contrib/clarknet-http.html.

[4] X. Chen and P. Mohapatra, "Lifetime behavior and its impact on Web caching," in *IEEE Workshop on Internet Applications (WIAPP'99)* , (San Jose, CA), July 1999.

[5] Y. Hu, A. Nanda, and Q. Yang, "Measurement, analysis and performance improvement of the apache web server," in *18th IEEE International Performance, Computing and Communications Conference (IPCCC'99)*, February 1999.

[6] J. L. Hellerstein, "An approach to selecting metrics for detecting performance problems in information systems," in *Proceedings of the ACM SIGMETRICS conference on Measurement & modeling of computer systems*, May 1996.