Universidade Federal do Paraná
Departamento de Informática

Cristina Duarte Murta
Tarcísio Paulo Corlassoli

Fastest Connection First: A New Scheduling Policy for Web Servers

Technical Report
RT_DINF 002/2002

Curitiba, PR
2002

Fastest Connection First: A New Scheduling Policy for Web Servers

Cristina Duarte Murta[1]
cristina@inf.ufpr.br

Tarcísio Paulo Corlassoli[2]
tpc@pb.cefetpr.br

[1]Department of Computer Science - Federal University of Paraná
POBox 19081 - 81531990 Curitiba PR Brazil

[2]Departamento de Informática - Centro Federal de Educação Tecnológica do Paraná
Via do conhecimento, Pato Branco, Paraná, Brazil

Abstract

Scheduling policies for Web servers have been designed to improve the delay at the servers, which may be an important part of the response time. However, these policies have not taken into consideration the interaction between the server system and the host TCP implementation. Web servers are used mainly in the wide-area Internet, which presents high variability in bandwidths, round-trip times and packet loss characteristics.

This paper introduces a new scheduling policy for the processing of the static HTTP requests in Web servers. This policy, called FCF (Fastest Connection First), gives priority to HTTP requests based on the size of the requested file and on the throughput of the user's connection. The requests for smaller files issued through faster connections receive the higher priorities. Our goal is to improve the mean response time. If the fastest connections receive priority at the server, they will have all good conditions (in the server and in the network) to finish quickly, contributing to minimize the mean response time. This policy takes into consideration the interaction between the server system and the TCP implementation at the server. We compare FCF with FIFO and SRPT policies. We find that FCF gives the best mean response time, although SRPT provides the shortest server delay. We conclude that the scheduling policy should be aware of the wan conditions and take them into consideration for giving priorities to the requests.

Keywords: web server, scheduling policy, web performance

1. Introduction

Web servers service requests from a large and heterogeneous user population. The diversity of the user characteristics is showed not only by their culture, idiom and geographical differences but also by their hardware and software platforms and their connectivity conditions. The connectivity of the Web users presents high variability. The link capacity goes from a 14.4 Kbps modem to a 10 Mbps connection [8]. In addition to the high variability of the capacity of the user's connections, some dynamic factors contribute to increase the variability of the connection speed such as congestion, number of hops in the TCP connection path, network and router delays, bottleneck links and many characteristics of the TCP implementations.

Scheduling policies for Web servers have been designed to improve the delay at the servers, which

may be an important part of the response time. It has been shown that the performance of the Web servers is affected by the Internet traffic [5, 17, 18]. In spite of these work, proposed scheduling policies for web servers do not take into consideration the interaction between the server system and the host TCP implementation which handles the TCP connections at the server. Previous work [11, 12, 13] has shown that when the size of the file requested is the main criteria for the scheduling policies, very good improvements in the server response time are observed in relation to policies that make no use of this information. In fact, it has long been known that SRPT (Shortest Remaining Processing Time) has the best mean response time of all scheduling policies [7,11]. Our work differs from [12] because our scheduling is mainly based on the throughput of the user connection while theirs is based only on the size of the response.

This paper proposes a new scheduling policy for Web servers that explores the high variability of the Web, observed in two of its characteristics: the speed of the connections and the response file sizes. The new idea is to explore the variability of the users connectivity. The proposed policy gives the highest priority to the request issued through the connection that will be completed first, that is, the connection that will have the smallest connection time. The fastest connection is the one that has requested the smallest document and has the best connectivity conditions (large bandwidth, low latency, no congestion). The proposed policy is called Fastest Connection First (FCF).

A policy that gives priority at the server to the requests issued through fast connections will reduce the service time of these requests and the life time of the respective connection. As a consequence, the number of pending requests is reduced. Pending requests are those which could have already been served but are not finished due to the transfer time (Internet time). When a connection is closed, the server resources reserved to that connection will be available for opening new connections. Our goal is to improve the mean response time seen at the user side. Our idea is to take into consideration the variability of the throughput of the opened connections. We think that we should give to the request the same level of service it experiments at the network. Our intuition is that if the connection has a high throughput, we should give high priority to the request from this connection, giving the request at the server the same conditions that the connection experiments at the network. This policy can be applied not only for Web servers but also for Web caches and other servers that experiment the same variability in those characteristics.

When a server gives priority to the users with the best connectivity conditions, it is helping to keep the response time according to the user expectation. The users connected by very fast connections want to experiment very small response times. Using FCF the request is served as fast as throughput of its respective connection. The main issue of our policy is to assure that the buffers of the faster connections will receive data quickly from the file system. The user with very good connectivity conditions might perceive the delay at the server when the server is overloaded. The service time is not perceived by users of not so fast connections.

To show the effectiveness of our proposal, we simulated FCF and compared it to SRPT and FIFO policies. We built simulation models for the server and the Internet. The workload was modeled using SURGE [3, 14]. The server model includes CPU, disk and network interface resources. The TCP and HTTP protocols were modeled, including RTT, congestion and packet loss characteristics. We find that FCF gives the best mean response time, although SRPT provides the shortest server delay. It means that the scheduling policy should be aware of the wan and connection conditions and should take them into consideration for giving priorities to the requests.

The rest of this paper is organized as follows. Section 2 describes the Fastest Connection First (FCF) scheduling policy. Section 3 discusses the experiment of simulation designed to compare the performance results of FCF, FIFO and SRPT policies. Section 4 presents the performance results.

Section 5 concludes the paper and briefly discusses plans for future work.

2. The Fastest Connection First (FCF) Scheduling Policy

Web servers service many requests concurrently. For example, the Apache Web server is a process-based server that forks several processes and serially accepts new connections. It makes the admission of the requests according to FIFO policy, no matter the response size or other request characteristic. So, the process that will get a system resource (CPU, disk or network interface) is chosen according to the arrival order.

FCF policy gives priority to the processes according to some characteristics of the specific request issued. The next process to get the resource is the one with the highest priority. The priority is based on two parameters: the throughput of the connection and the number of bytes to be processed. The idea is to give priority to the request which connection can be finished first, that is, the smallest request issued through the fastest connection (the one with the best transmission rate). The transfer rate depends on the connectivity conditions of the user as well as the dynamic behavior of the Internet.

In this work we make the assumption that the time required to serve a request is well approximated by the size of the response (size of the requested file). This is quite reasonable when the request is static, that is, when all the work the server has to do is to read a file at the disk and serves it. Static requests are the majority in a Web server workload.

The FCF policy algorithm has two steps. First it searches the request queue for the request with the shortest remaining number of bytes to process. Second, it searches the queue again for all requests which size is smaller than the smallest one times some constant (called beta). From those, we choose the one with the highest estimated throughput. Beta defines the range of the second search and the trade-off between size and throughput. If beta is large we are giving priority to throughput. On the other side, if beta is small (close to 1), we are giving priority to the remaining size of the request. In the next section we describe the process to estimate the throughput.

This policy aims to consider the connectivity conditions of the user, including all aspects that affects the data transmission in the Internet. At the end, we are considering the end-to-end performance of each connection and giving priority to the one that has the best performance. Fast connections tend to have smaller RTT and larger TCP reception and congestion windows [9]. As a consequence, if priority is given to the requests issued through fast connections, at the server, these requests will be able to finish quickly, releasing the resources at the server to the requests waiting in the queue.

Web requests are served through a TCP connection. Due to the TCP congestion control (slow start) a small number of packets are sent each time, while the remaining packets must wait for the ack confirmation package [6,10]. The acks from the fast connections arrive first and then new packets can be sent through these connections. The FCF policy aims to ensure that the socket buffers of the fast connections will always have data to be sent as soon as the ack arrives. It means that the next packets will be ready to be sent. FCF policy looks forward to handling the requests at the server according to the performance level of the respective connection. The performance of the connection is taken by the measured throughput of the connection. The higher the throughput of a connection, the higher the priority of the respective request at the server.

In order to have the packets ready to be sent, the server must give priority to the requests at the CPU, at the disk and at the network interface. The buffers of the fast connections must receive the

data from the disk as soon as the acks arrive. When the load on the server is light or moderate the connection buffers usually have the data to send as the ack arrives [16] no matter the scheduling policy. Nevertheless, when the server is overloaded, giving priority to the small requests issued through fast connections will ensure that the buffers of these requests will have data to transfer when the acks arrive. In addition, as the server load grows the TCP control messages and IP datagrams queues become longer. The FCF policy may be applied to these queues as well [23].

3. The Simulation Model

The FCF policy was simulated and compared to two other policies, FIFO (First In First Out) and SRPT (Shortest Remaining Processing Time). The simulation model includes the server model, the workload generator and the Internet model. We describe the simulation model and the experimental methodology in the next sections.

3.1. Server Modeling

Static requests make use of three system resources: CPU, disk and network card. The processing of the requests at the server can be describe as: opening of a TCP connection, getting the HTTP request, parsing the HTTP request header, reading the file in the disk, package and sending the data through the connection, close the connection and make the register at the log file.

The service demand of a request is composed of three parts [5, 7, 12]: the CPU time, that represents the time spent at the CPU to parser the request, to pack and to unpack the TCP segments and IP datagrams, to control the data transfer from the disk to the socket buffer and then to the network interface, and other demands to handle the TCP connection. The second part is the time to transfer the data from the disk to the socket buffer. The last part is the time to transfer the data to the network interface.

These demands can be associated with each resource of the system, CPU, disk, and network card. The Web server simulation model is composed by three queues, one for each resource. The queues hold HTTP requests. There is a socket buffer corresponding to each connection. Each queue is driven by a scheduling policy. The CPU queue was implemented as a FIFO queue because the information needed to give the priority according to FCF policy was not available before the parser of the request. The requests admitted at the CPU are serviced according to PS (Processor Sharing). The requests at the disk queue are waiting for data transfer from the file system to the socket buffer. The data is read in blocks of 16 Kbytes (block mode). The network queue transfer the data from the socket buffer to the network interface buffer and closes the connection when the transfer has finished. Otherwise, it will place the connection into the disk queue. Thus the connections alternate between the disk and network queues until all the requested data has been served.

3.2. Server Workload Modeling

The second component of the simulation model is the server workload. To generate the server workload we used the SURGE benchmark [3, 14]. SURGE has an workload generator designed to reproduce important characteristics of Web workloads including heavy-tailed file size and request distribution, Zipf popularity and temporal locality of the requested files, distribution of the embedded references, on-off times and the number of simultaneous users. SURGE generates requests simulating the access of hundreds or even thousands users.

SURGE defines the concept of user equivalents (UE). A user equivalent is a single process which

represents the load produced by a single user. The workload intensity in SURGE is given by the number of user equivalents. In our simulation, the UE ranges from 70 to 700.

3.3. Internet Modeling

Modeling the Internet is a difficult task [21, 22]. Nevertheless, many works have been modeled and studied some aspects of the Internet [2, 9, 10, 19, 20]. Despite this work the Web server performance evaluation has been done on high-speed LANs.The Internet modeling in our work considers performance aspects of WANs. In our work the Internet was modeled considering the main aspects of the TCP and the HTTP/1.1 protocols. We focus on the following characteristics: RTT, bandwidth ranges, network traffic and losses. We discuss each of these characterizations.

The first characterization is of RTT and bandwidth. Table 1 presents the probability distribution of the RTTs and of the bandwidths based on work [9]. We defined four classes of minimum RTT: 20, 50, 90 and 280 ms. Then we defined the bandwidth distribution and the corresponding RTT class. A more realistic distribution of the RTTs must consider the effects of the traffic. We model the RTT oscillations according to a limited Pareto distribution, having the minimum RTT as a parameter for the minimum value. The maximum value was set up as the minimum times eight. The alpha parameter was 1.4. The set of values generated by this model is according to the statistics of observed values for RTT presented in [2]. Packet losses were defined in 5%.

The TCP and HTTP/1.1 protocols were modeled in our simulation. The model included the TCP connection establishment and termination, slow start, flow control and congestion control (advertised and congestion windows). The mechanisms for fast retransmit and recovery were not modeled. The calculation of the transmission rate (throughput) needed to attribute the priorities in the FCF policy is made by the simulation model.

Bandwidth	Percentage	RTT class
14 Kbps	2	90 and 280
28 Kbps	4	90 and 280
33 Kbps	12	90 and 280
56 Kbps	34	90 and 280
128 Kbps	19	20 and 50
1 Mbps	13	20 and 50
4 Mbps	9	20 and 50
10 Mbps	7	20 and 50

Table 1 : Bandwidth distribution and RTT classes.

4. Simulation Results

In this section we summarize our results. The simulation goal is to evaluate and compare the performance of the policies FCF, FIFO and SRPT. The performance metrics are the delay at the server, the response time seen by the client, the slowdown and the analysis of starvation of large requests and of slow connections. First we present the workload characterization, followed by the performance results.

The results for FIFO run means that this policy was applied to the three queues (CPU, disk and

network interface). The FCF option means that the FIFO policy was applied to CPU queue and the FCF policy was applied to the other queues. The SRPT option means that the FIFO policy was also applied to CPU queue and the SRPT policy was applied to the other queues.

The beta constant value was two in the simulation. It means that the files which size was up to two times the smallest file were considered to get the highest priority. In other words, the highest priority was given to the request issued through the fastest connection (highest throughput) from the requests which size is up to two times the smallest response.

It is important to highlight that some results for FCF and SRPT are similar because those policies share the same criteria, the response size. Both give priority to the shortest remaining processing time. The difference between them is that FCF also consider the throughput of the user's connection. The goal of FCF is to help the interaction between the server and the network. As already known, SRPT is the best policy when we consider the metrics related to the server [11] such as server delay. FCF improves the request response time measured at the client side. We argue that the user response time is the most important metric and should be the one chose to be improved. As we will show, FCF presents the best results for response time.

4.1 Server Workload Characterization

The server workload was generated by SURGE. Each simulation run was composed by more than a million requests. The mean file size was 14 KB. The UEs varying from 70 to 700. From the server point of view, each run serviced the same requests in the same sequence. From the user point of view, some users may have issued a different number of requests according to the priority given to them by the scheduling policy. This behavior is due to the SURGE generator. As soon as the user receives the file requested, he is able to issue another request. So the users with the higher priorities will get the responses quickly and will send more requests.

The workload was classified in eight classes according to the response (file) size. Figure 1 shows a plot with the distribution of the bytes and of the requests for each class. Most of the requests are for small files while most of the bytes are requested by a small set of requests. More than 90% of the requests are for files smaller than 32 KB and, on the other side, only 1.47% of the requests are for files larger than 128KB. The requests for files larger than 128 KB compose almost 40% of the server workload. This distribution of the bytes and files represents a real Web server workload.

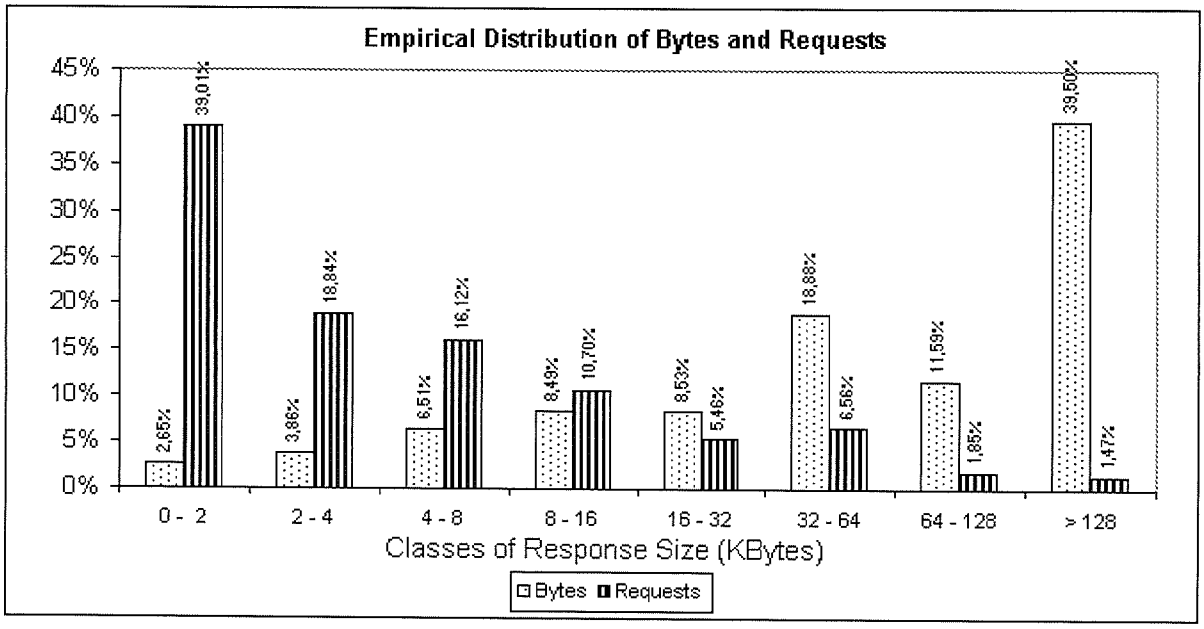

Figure 1: Distribution of the requests and bytes in the classes based on the file sizes.

4.2. Analysis of the Starvation of the Large Files

A common concern about the SRTP is its unfairness. Since it gives preference to small files, large ones may starve. We can observe in the plot of Figure 1 that about 85% of the requests are for files smaller than 16 KB and they represent 21,5% of the server demand. If the server gives priority to the small requests, it will have 78,5 % of the resource to handle the last 15% (larger) requests. Considering these numbers, it is not necessary to have a dynamic criteria for the scheduling policy to avoid starvation because most of the server resources will be dedicated to the large requests. This approach is also taken in a number of works [11, 12, 13].

Figure 2 shows the response time for each class of file size for a workload of 700 UE. We observe that the class with files larger than 128 KB experiments a small penalization when compared to the size-independent (FIFO) policy. This class comprises 1.47% of the requests only. We see that for more than 98% of the requests there is a considerable improvement in the mean response size given by FCF and SRPT policies compared to FIFO.

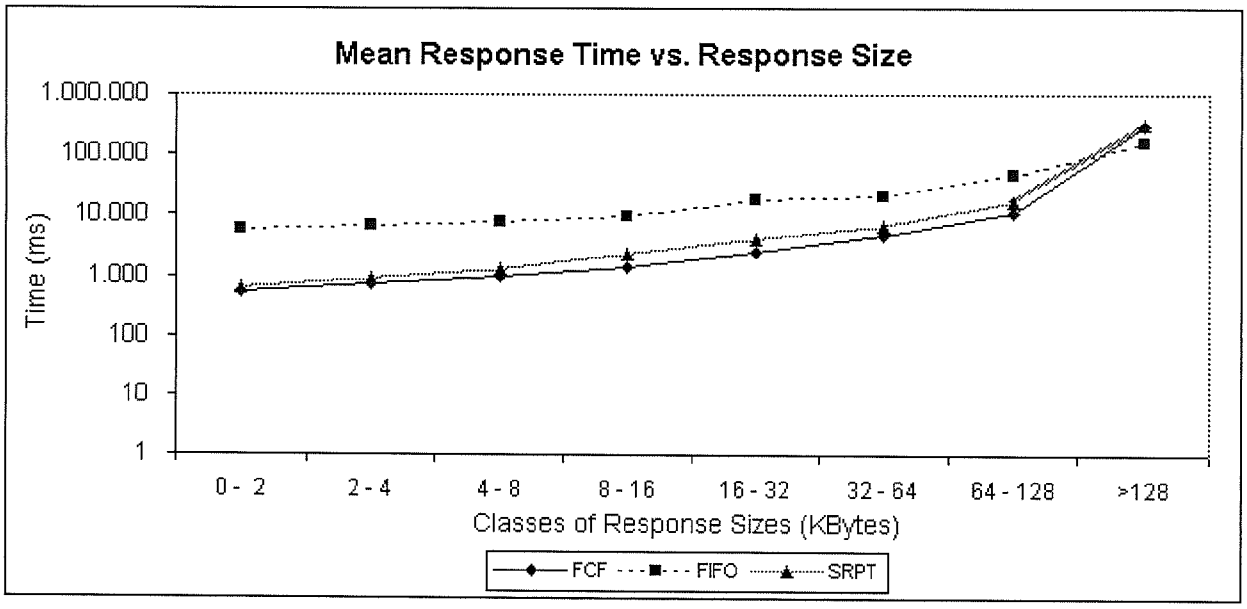

Figure 2: Mean response time per class of file sizes for workload of 700 UEs.

4.3 Analysis of the Starvation of the Slow Connections

The plot on Figure 3 shows the mean response time seen by the client as a function of the connection bandwidth. The question is weather the slow connections suffers. We observe that the response times are proportional to the classes of connection bandwidth for all policies. The higher the bandwidth, the smaller the response time. This result validates the TCP simulation. It means that the simulated TCP algorithms recognizes and controls the flows.

FIFO has the worst response times for all connectivity conditions. It means that there is no starvation of the connections with low throughput. FCF results are worse than SRPT results when the bandwidth is low. FCF has the best results for the best connectivity conditions. We also conclude that the FCF policy interacts well with the TCP algorithms, leading to improvements in TCP performance in the case of good connectivity conditions. This results also shows that we have reached our goal that was to improve the response time for the fastest connections.

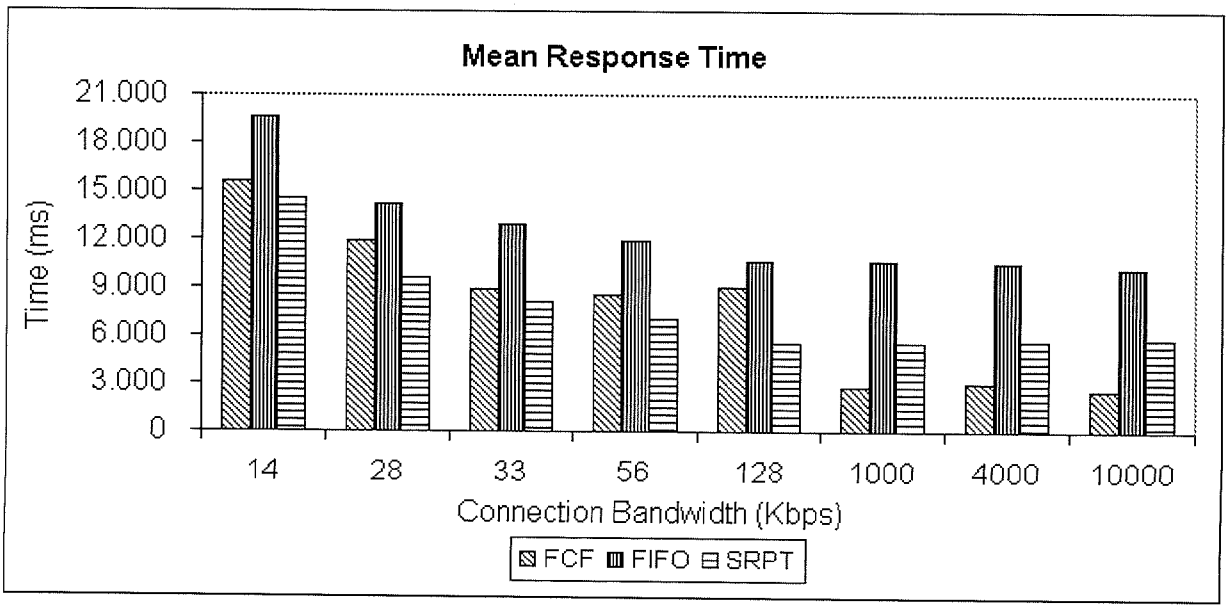

Figure 3: Mean response time as a function of the connection bandwidth.

4.4 Slowdown

The slowdown evaluates the fairness of the policies. It is calculated as the waiting time in all queues divided by the total service time. A small value for slowdown means that the waiting time the tasks experiment is proportional to their size. Figure 4 shows the slowdown for the three policies. We see that FIFO discipline is much more unfair than FCF and SRPT policies.

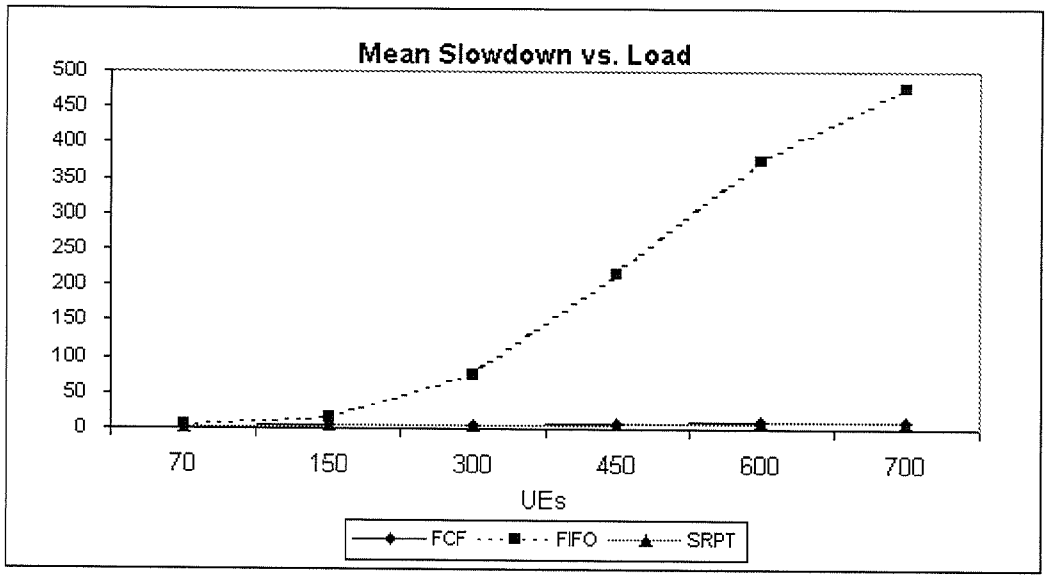

Figure 4: Slowdown.

4.5 The Delay at the Server

Figure 5 shows the mean delay at the server as the load increases. The delay is the sum of the service demand at all resources plus the waiting times at the server queues. FIFO has the worst result. FCF and SRPT have similar results but consistently better for SRPT. This is an predicted result since SRPT discipline is known to have the best server delay of all queueing disciplines. So, this result also contributes to validate our simulation. We observed that FCF also get good results at the server because it considers the size of the request as a criteria to give priorities.

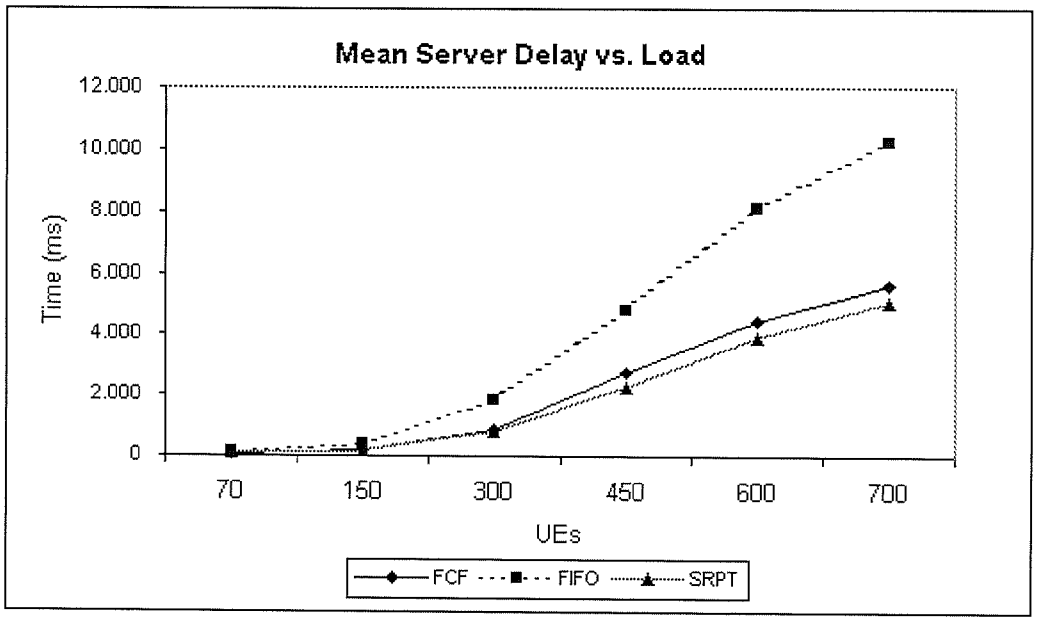

Figure 5: Mean Server Delay

4.6 Mean Response Time

Figure 6 shows the result for the most important metric, the mean response time, that is, the delay seen at the client. FIFO has again the worst result. FCF has consistently the best results at all loads. Although FCF policy has not the best server delay, the interaction done with the server and the TCP implementation leads to the best results in response time. The response time includes the server delay and the transmission time. We did not model the delay at the client.

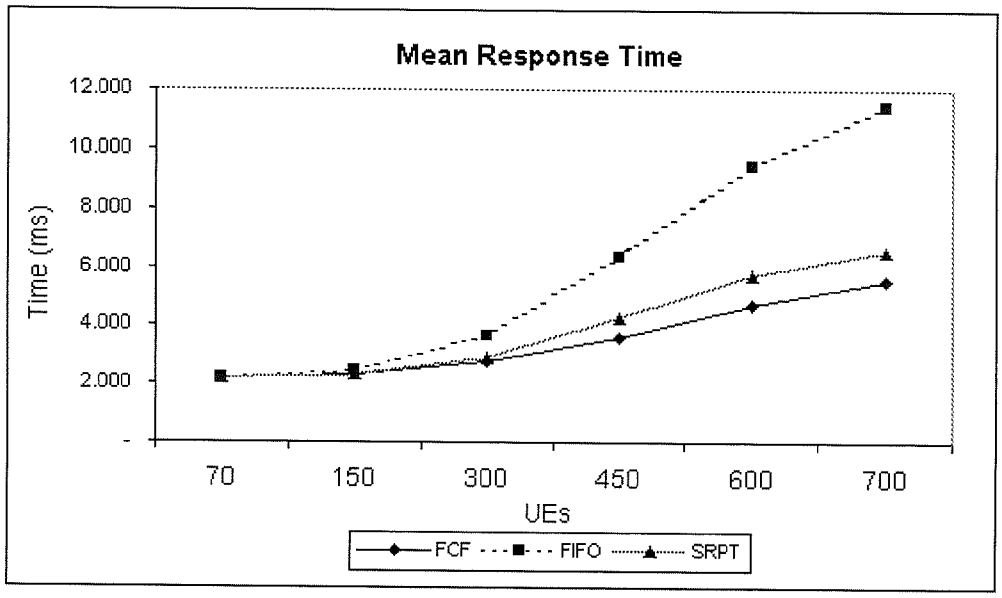

Figure: Mean Response Time

6. Conclusion and Future Work

The information sharing between a Web server system and the TCP connections opened in the

server are explored in this paper to propose a new scheduling policy that is based on the throughput of the user's connection. The proposal policy, FCF improves the mean response time with no excessive penalization for large requests and slow connections. The results show evidences that the different levels of connectivity in the Internet affect the performance of the Web server, and that this information can be used to improve the performance of the system significantly.

Why should this policy be taken into consideration? FCF improves the interaction between server and network systems, giving similar priorities to the requests in both environments, and leading to better mean response time. In the long term, even as bandwidth costs continue to drop and higher end-user speeds become available, FCF wil continue to reap the benefits because a bunch of facts. First, non-uniform bandwidth and network latencies will persist (or even be higher) due to financial constraints and physical limitations such as environment and location. Second, network distances and latencies are increasing. Third, user expectations of faster performance continue to increase. In fact, we think that the variability of the bandwidth and latencies will increase because although new high-end speeds become available, many users do not improve their connectivity conditions. So, the range of the speeds increases, and also increases the variability.

References

[01] ALMEIDA, J.; DABU, M.; MANIKUTTY, A.; CAO, P. **Providing Differentiated Levels of Service in Web Content Hosting.** In Proceedings of Workshop on Internet Server Performance, Madison, Wisconsin, June 1998.

[02] CROVELLA, M. E.; CARTER, R. **Dynamic Server Selection in the Internet**. In Proceedings of HPCS'95: Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, August 1995.

[03] BARFORD, P.; CROVELLA, M. **Generating Representative Web Workloads for Network and Server Performance Evaluation**. In Proceedings of ACM SIGMETRICS'98, Madison, July 1998.

[04] ALMEIDA, V. A. F.; ALMEIDA, J; YATES, D. **WebMonitor: a Tool for Measuring World-Wide Web Server Performance**. In Proceedings of Seventh IFIP Conference on High Performance Networking, White Plains, NY, April 1997.

[05] DILLEY, J.; FRIEDRICH, R.; JIN, T.; ROLIA, J. **Measurement Tools and Modeling Techniques for Evaluating Web Server Performance**. In Proceedings of 9th Int. Conf. on Modeling Techniques and Tools, vol. 1245 of Lecture Notes in Computer Science, p. 155-168. Springer-Verlag, 1997.

[06] MURTA, C. D.; ALMEIDA, J.; ALMEIDA, V. A. F. **Performance Analysis of a WWW Server**. In Proceedings of 22nd International Conference on Technology Management and Performance Evaluation of Enterprise-Wide Information Systems, San Diego, California, December 8-13, 1996.

[07] CHERKASOVA, L. **Scheduling Strategy to Improve Response Time for Web Applications**. In Proceedings of High-performance Computing and Networking: International Conference and Exhibition, p. 305-314, 1998.

[08] Tenth WWW User Survey (Conducted October 1998), Graphics, Visualization & Usability (GVU) Center at Georgia Tech. http://www.gvu.gatech.edu/u ser_surveys.

[09] ACHARYA, A.; SALTZ, J. **A Study of Internet Round-Trip Delay**. Technical Report CS-TR-3736, Department of Computer Science, University of Maryland, USA, December 1996.

[10] CARDWELL, N.; SAVAGE, S.; ANDERSON, T. **Modeling the Performance of Short**

TCP Connections. Technical Report, Department of Computer Science and Engineering, Univ. of Washington, November 1998.

[11] HARCHOLBALTER, M.; CROVELLA, M.; PARK, S. **The Case for SRPT Scheduling in Web Servers**. Technical Report MIT-LCS-TR-767, MIT Lab for Computer Science, October 1998.

[12] CROVELLA, M.; FRANGIOSO, B.; HARCHOL-BALTER, M. **Connection Scheduling in Web Servers**. In Proceedings of *USITS '99: USENIX Symposium on Internet Technologies and Systems*, p. 243-254, Boulder, Colorado, October 1999.

[13] HARCHOL-BALTER, M.; BANSAL, N.; SCHROEDER, B.; AGRAWAL, M. **Implementation of SRPT Scheduling in Web Servers.** Technical Report number CMU-CS-00-170. Carnegie Mellon School of Computer Science, October 2000.

[14] BARFORD, P.; CROVELLA, M. **An Architecture for a WWW Workload Generator.** In Wide Web Consortium Workshop on Workload Characterization, October 1997.

[15] BESTAVROS, A.; KATAGAI, N.; LONDOÑO, J. **Admission Control and Scheduling for High-Performance WWW Servers.** Technical report BUCS- TR-97-015, Boston University, Computer Science Department, August 1997.

[16] DRUSCHEL, P.; BANGA, G. **Lazy Receiver Processing (LRP): A Network Subsystem Architecture for Server Systems**. In Proceedings of OSDI'96: Second Symposium on Operating Systems Design and Implementation, October 1996.

[17] EDWARD, S. S.; SUTARIA, J. **A Study of the Effects of Context Switching and Caching on HTTP Server Performance**. http://www.eecs.harvard.edu/stuart/Tarantula/FirstPaper.html.

[18] BANGA, G.; DRUSCHEL, P. **Measuring the Capacity of a Web Server Under Realistic Loads**. In World Wide Web Journal (Special Issue on World Wide Web Characterization and Performance Evaluation), 1999.

[19] BARFORD, P.; CROVELLA, M. **Critical Path Analysis of TCP Transactions.** In Proceedings of ACM SIGCOMM'2001: Special Interest Group on Computer Communication, Stockholm, Sweden, September 2000.

[20] BARFORD, P.; CROVELLA, M. **Measuring Web Performance in the Wide Area**. In Proceedings of Performance Evaluation Review, August, 1999.

[21] FLOYD, S.; PAXSON, V. **Why We Don't Know How to Simulate the Internet**. In Proceedings of Winter Simulation Conference, Atlanta, GA, December 1997.

[22] FALOUTSOS, M.; FALOUTSOS, P.; FALOUTSOS, C. **On Power-Law Relationships of the Internet Topology**. In Proceedings of ACM SIGCOMM '99: Special Interest Group on Computer Communication, p. 251-262, August 1999.

[23] DRUSCHEL, P.; BANGA, G.; MOGUL, J. C. **A Scalable and Explicit Event Delivery Mechanism for UNIX**. In Proceedings of USENIX Annual Technical Conference, Monterey, California, June 6-11, 1999.