CATP: A Context-Aware Transportation Protocol for HTTP*

Huamin Chen and Prasant Mohapatra
Department of Computer Science
University of California, Davis, CA 95616.
Email: {chenhua, prasant}@cs.ucdavis.edu.

Abstract

The rendering mechanism used in Web browsers have a significant impact on the user behavior and delay tolerance of retrieval. The head-of-line blocking phenomena prevents the browser to render partial results within the HTTP document. This phenomena stems from two factors: TCP's in-order data uploading to the browsers and HTML tag matching constraint. We propose a context-aware transportation protocol (CATP) to run on top of UDP. This protocol does not transmit HTML pages in-order. Instead, it reorganizes the pages and transmit HTML tags first before transporting their enclosed data. Conforming browsers receive the page structures and fill them in with subsequent data packets in whatever sequence they arrive. As a result, lost and delayed packets do not hinder rendering of those that are logically behind but have already arrived at the client sides. Thus the retransmission of the lost frames can be concealed and overall user perceived performance improved. The user-perceivable performance is quantified in terms of silent time during which no activity is observed at the browser display. The protocol also facilitates partial content caching, non-interactive applications of Web services. We validated this protocol through prototype implementation and compared the performance with TCP and in-order delivery UDP schemes. Our protocol provides better user-perceivable performance under various loss loss rates and document sizes.

Keywords*: HTTP, Transportation protocol, UDP, head-of-line blocking, out-of-order rendering.*

1 Introduction

Web traffic is growing at a fast pace and becoming a dominant component of Internet traffic. Web server performance has been a hot research topic. The widely used metrics to assess web server performance are server throughput and transaction latency. The server throughput measures the number of requests or bytes serviced in a certain time quantum. While the transaction latency refers to the time interval between when a request sent to the server and the response's arrival at the end user. We argue that the two metrics are not sufficient to assess the user perceived performance. In [4], we justified that server throughput is not an appropriate measurement in session-based web traffic like e-commerce. Similarly transaction latency cannot capture the user's reaction as well. For instance, consider two scenarios of transmitting of a web page with 30 seconds latency. In the first case, the end user receives new objects in the Web page every 3 seconds and every 10 seconds in the second case. Although the total time latency is the same, the end user in the second case perceives a more stagnant transmission thus worse perceived performance. Furthermore, larger silent time may lead to request abortion by the clients. We define *silent time* as the time interval during which no new data are fed to the end user. We are therefore motivated to investigate how to curtail the silent time to provide better user-perceived performance in congested situations. Smaller silent time will be perceived as continual flow of information by the user, and thus the overall latency of the page rendering will be hidden in terms of perception.

The current web infrastructure is built on top of TCP, which provides reliable delivery between communication peers. Though TCP has been successful in delivering web traffic in many environments, previous research work have revealed the following intrinsic problems with TCP that hinders further performance improvement [6]. TCP is a connection oriented protocol that ensures communication reliability. The participants first setup a communication channel by a three-way handshaking process, which is lengthy for high delay environment and short connections like HTML page delivery. Web servers could redirect the incoming requests to others because either the requested contents have moved or due to some load balancing policies employed to distribute load among multiple servers. The clients thus have to setup new connections. During the process of transmission, the receiver must send ACKs to notify successful

*This research was supported in part by the National Science Foundation through the grants CCR-0296070 and ANI-0296034.

arrivals and update the sender's TCP send window. Since the processing of ACKs is interrupt driven, large numbers of simultaneous ACKs could overwhelm the servers and erode their resources in processing more valuable tasks. If some data frames are lost during the transmission, the receivers have to wait for retransmission, which will not be launched until 3 duplicate ACKs successfully reach the sender or certain timeout event occurs. During this period, the receiver cannot process the out-of-order frames. As a result, the end users perceive inactivity in page rendering: a long silent time before new contents are displayed. This phenomena is called head-of-line (HOL) blocking.

We investigated the maximum silent time in a daily traffic trace obtained from [12]. We studied several days of the HTTP tcpdump traces on the 18Mbps trans-Pacific link. We analyzed the difference between maximum silent time of each connection and the corresponding inter-packet arrival time. The inter-packet arrival time characterizes the network transmission latency. So the difference tells how other factors like packet loss and reordering contribute to the silent time. Out of 17,158 completed HTTP connections, 2,083 were found to have such difference. The cumulative distribution of the difference is plotted in Figure 1. The average value is 8.6 seconds, which is beyond the end users' tolerance latency of 8 to 10 seconds [9, 10, 2]. It is also found in the trace that, silent time is independent of the file sizes. Thus it is likely that even during downloading small files the end users could experience lengthy silent time.

Silent time due to HOL blocking stems from the packet drops in congested routing devices, packet loss in unreliable environment like wireless network, and packet reordering. Reloading a web page usually opens a new TCP connection that knows nothing about the previous one and cannot reuse the out-of-order packets that have successfully arrived. As a result, it could only exacerbate the already strained routers. In an inherent lossy wireless network, reload does not guarantee the successful reception of the entire page. Packet reordering, as concluded in [1], which could profoundly affect TCP performance, becomes increasingly serious in DiffServ environment where packets belonging to the same connection can enter different forwarding queues.

We believe that the solutions to this problem need to incorporate two indispensable factors. First, the receiver can reuse the partial results so that the reload of the same page does not require retransmission of these data. Obviously, TCP is incapable of this functionality, since TCP allows no sharing of data between connections. We were thus motivated to use UDP as our transportation protocol. Second, there should be minimal dependency between the packets so the loss or delay of one packet does not affect the rendering of others. Using this regard, there will always be inputs to the users' perception thus the silent time is amortized.

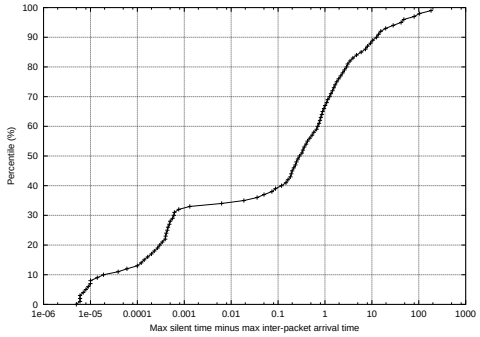

Figure 1. Cumulative distribution of maximum silent time minus maximum inter-packet arrival time.

This is also an instantiation of what was suggested in [13] to provide feedback in form of a progress indicator.

On the other hand, the constraint of tag matching in rendering Web pages also contributes to the silent time. Insufficient context information, especially rendering styles, can generate significant silent time for users using low speed links. Early delivery of context information can resolve this abnormality.

We believe that the average size of Web pages increases with the wide usage of authoring and content management tools for better presentation and management. The XML documents that are used in data intensive applications are expected to be even bigger. Low speed links like dial-up will continue to be a significant means of network connections [8]. As a result, silent time due to transmission of large Web pages on slow links will remain to be outstanding.

In this paper, we proposed a context-aware transportation protocol (CATP) for web traffic on top of UDP. In CATP, the HTML page contents are fragmented into several frames. The first frame (index frame) contains HTML tags and pointers to their enclosed data. The transmission of index frame is protected against loss. Subsequent frames (data frame) contains data that can be referenced by the pointers. The content in the data frames are self-contained in syntax and can be rendered for presentation without the help of other frames thus eliminating HOL blocking. CATP also facilitates caching, content adaptation and parallel downloading from multiple servers. We have implemented the protocol in an Apache web server and evaluated its performance. The experimental results demonstrate that the silent time is significantly amortized using the proposed scheme as compared to pure UDP and TCP under various link and document characteristics.

The rest of the paper is organized in this way. Section 2 discusses the transmission problems of TCP and UDP and presents CATP. The experimental evaluation is in Section 3 followed by the related works are discussed in Section 4. At

last are the concluding remarks Section 5.

2 Context-Aware Transportation Protocol

It is obvious that if the application can handle out-of-order data, not only the silent time is amortized but also the packet retransmission could be concealed in the rendering process. Since TCP only delivers in-order data streams to applications, we chose UDP as our transportation protocol. In addition to TCP's in-order delivery, another factor that prohibits out-of-order rendering is the HTML tag. The start and end of the well-formed HTML tags must match to make their enclosed data meaningful. The tags are essentially predicative that establish the partial order of content within the page. Because tags can be nested, the interpretation of HTML files has to be sequential. Consider a HTML paragraph, <H1>ABCD</H1> <H2>EFGH</H2> <H1>IJKL</H1>. Assume the paragraph is fragmented into 5 segments: <H1>ABCD, </H1><H2>, EFGH, </H2><H1>, and IJKL</H1>. Suppose these segments are transported on UDP and arrive out of order and the receiver receives segment 1, 3 and 5 first, he could mistakenly think EFGH should be rendered using style H1! In this case, in order to accurately render EFGH, the receivers has to wait for the arrival of the associated tags. It is conceivable that if the tags are separated far apart, loss or delay of any data segments that are enclosed by the tags will postpone the rendering of the others.

We have conducted a survey of the *tag distance* in over 20 web sites including MSNBC, CNN, YAHOO, AMAZON.COM. Tag distance refers to the size of content that is enclosed by a HTML tag pairs. For instance, the tag distance of HTML code <small>ABCD</small> is the string length of "ABCD" which is 4 bytes. We measured the tag distance in terms of number of the Ethernet packet size (~1500B). Obviously, tags with distance exceeding that size should be split into at least 2 packets. We only consider HTML 3.0 tags. Tags like <HTML>,<BODY>,<FRAME> are excluded in the measurement due to their minimal impact on page presentation. It is conceivable that in XML documents where tags proliferate the rendering stagnancy due to unmatched tags could be more serious.

We downloaded over 80,000 HTML files (including dynamic page outputs). The average file size of these files is 25.5KB. The cumulative distribution of the maximal tag distance in each file is plotted in Figure 2. Since x-axis is in log scale, distance 0 is omitted whose cumulative value is 55%. It is observed that, tag distance is nontrivial in the inspected pages; nearly half of these pages have to split tag pairs into several Ethernet frames and over 10% of them have tag distance more than 10 packets, which implicates that browsers could have to wait for 10 packets to completely interpret the rendering style of the enclosed content!

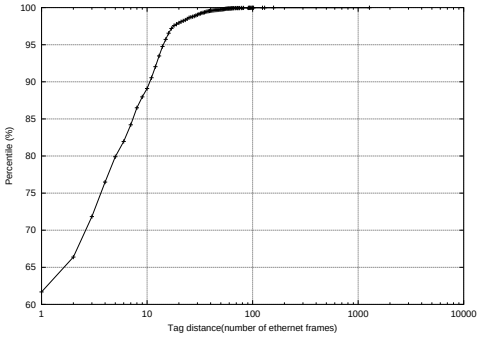

Figure 2. Cumulative distribution of tag distances.

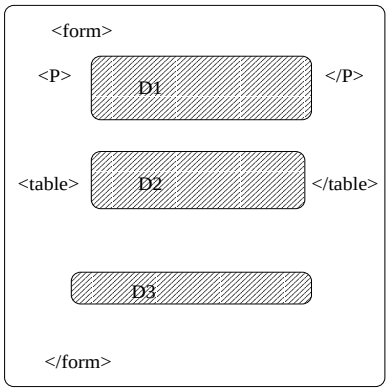

Figure 3. Sample HTML page.

Furthermore, loss or delay of any single frames within this range could leave the rest of them unrenderable.

The aggregation of tags is the skeleton or structure of the page. Usually, when tags are stripped off, content blocks enclosed by tag pairs are rank-less. We thus propose to separate HTML tags from their enclosed content and deliver them in different packets. We call this scheme context-aware transportation protocol (CATP). Our scheme is illustrated and compared in the following diagrams.

Figure 3 illustrates a HTML page with three data sets: paragraph D1, table D2, and control section D3 that includes submission buttons. The tag pairs <form>, and </form>, <table> and </table>, and determines the presentation style of their enclosed data sets. Consider the following possible transmission process illustrated in Figures 4 and 5 that could be encountered in current TCP protocol and in-order UDP delivery. CATP is shown in Figure 6 where pDi is the pointer to data set Di.

In Figure 4, the rendering of received data set cannot be done until the TCP layer receives in-order data. So if fragment <table>D2 is lost during transmission, the receiver's TCP layer will not transfer the fragment D3 to the browser.

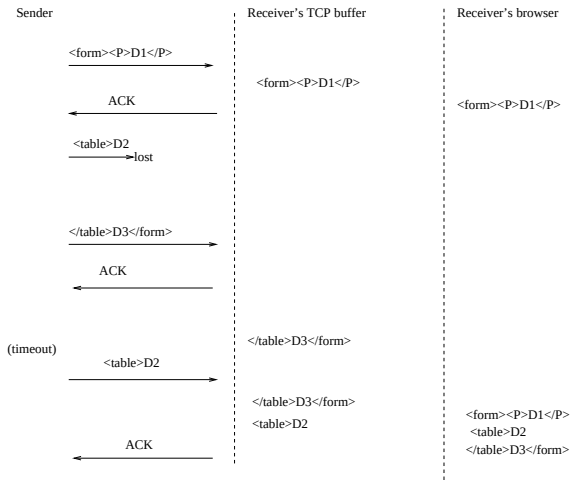

Figure 4. TCP transmission flow.

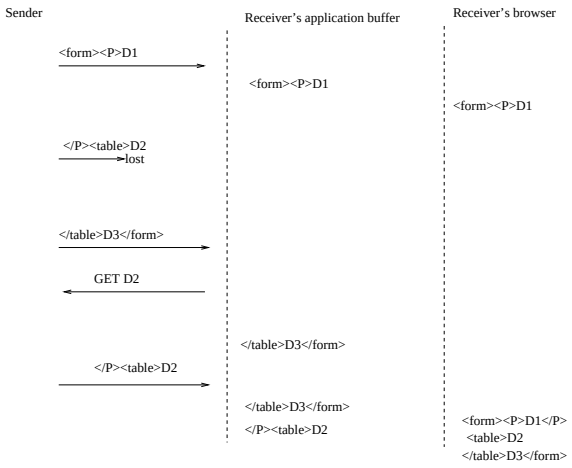

Figure 5. UDP transmission flow.

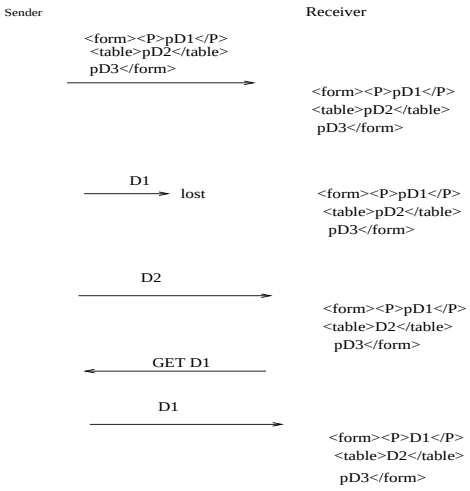

Figure 6. Out-of-order proof transmission flow.

The user has to wait for the retransmission of the lost segment. The silent time, denoted by T, in this occasion is determined by link loss and retransmission algorithm and if the sender uses TCP fast recovery algorithm, the silent time T is

$$\begin{aligned} T &= min(P(arrival\,of\,1\,packet) * TO, \\ &\quad P(arrival\,of\,3\,packets) * D) \\ &= min((1-p) * p^n * n * TO, \\ &\quad (\binom{n+3}{3} * p^n * (1-p)^3 * (n+3)) * D), \end{aligned}$$

where p is the packet loss probability, $n \in [0, \infty)$ is the number of lost packets, TO is the retransmission timeout value, and D is the link delay. The expected silent time is

$$E(T) = min(\frac{TO}{1-p}, (\frac{3+p}{(1-p)^2}) * D).$$

In the UDP enabled in-order delivery scheme illustrated in Figure 5, though the application can read out-of-order packets, it cannot correctly interpret the rendering style without the complete context information. Thus the loss of fragment </P><table>D2 leaves the third fragment (</table>D3</form>) with unmatched tags. The browser does not know whether the trailing tag </P> is in the lost packet or in the forthcoming packets, the rendering process cannot be done until its context information is complete. The silent time in this case is also a function of loss rate and retransmission algorithm.

In Figure 6, every transmitted packet is self-contained and not dependent on others. The index frame that contains HTML tags and pointers is sent first followed by the individual data packets. Upon arrival, the data packets fill in appropriate positions in the index frame. So the loss of any packet does not affect the rendering of those that have successfully arrived at the receiver side. Since the retransmission can be done in parallel with the transmission of other fragments, the silent time in this case is independent of the link loss rate and tag distance, thus the silent time due to packet loss or delay is

$$T = P(arrival\,of\,1\,packet) * D = (1-p) * p^n * n * D,$$

and the expected silent time is

$$E(T) = \frac{D}{1-p}.$$

3 Experimental Evaluation

3.1 Implementation

CATP was implemented in Apache 1.3.22 for Linux. The modification to original Apache distribution is small, less than 50 lines of original code were modified to support UDP and a new module was added to support page fragmentation delivery. We also implemented a UDP HTTP client to benchmark the performance.

The experiment was done on an emulated network environment with various offered link speed and loss rates by using NIST Net [11] on a Linux box with the kernel version 2.4.17. NIST Net is a Linux kernel module that emulates a WAN environment by adding time delay in packet transmission, controlling transmission rate, and constantly dropping packets. It is ideal to validate the effectiveness of our protocol in various conditions. Although NIST Net provides many network conditions, what interests us most is the variation of packet loss rates which is the major cause of HOL blocking in current Internet.

3.2 Experimental Results

Figure 7 plots the comparison of silent time of TCP, UDP and the proposed protocol under offered bandwidth 10KB/s to request a 10KB and a 50KB page with maximum of tag distances of 5 and 20 frames respectively. It is assumed there is no dependency among these data frames. For each protocol, 100 requests were initiated for each loss rate configuration and their average maximum silent time is plotted. In the TCP tests, the benchmarking tool ab reported operation timed out under high loss environment. So we could not collect the corresponding results under these conditions.

Figure 7 compares the impact of tag distance. It is observed that the large tag distance incurs longer silent time under all configurations. Note that TCP's performance is not directly affected by tag distances, but longer tag distance usually means larger file to transfer where TCP is more sensitive to loss. When the loss rate is low, TCP outperforms both UDP and our scheme due to its aggressive utilization of available bandwidth. But the advantage is very small. As the loss rate increases, TCP's silent time grows dramatically whereas our context-aware scheme remains relatively stable. Two factors are responsible for this phenomena. First, the retransmission in TCP stymies new packets arriving at the receiver thus extends the silent time. Second, upon packet loss, the TCP's congestion avoidance mechanism cuts the congestion window in half and decreases the transmission rate. These factors are absent in UDP which transmits data in constant rate. Thus the performance of UDP is better than TCP under highly lossy situations. But as discussed earlier, pure UDP does not circumvent the unmatched tag blocking thus is still subject to HOL blocking. Whereas the context-aware scheme is not obstructed by tags and thus exhibit even better performance.

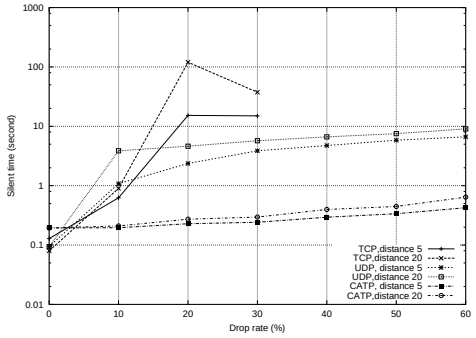

Figure 7. Tag distances are 5 and 20 frames

4 Related Works

Research on HTTP transportation protocols mainly focused on how to adapt TCP to transfer short HTTP files. Persistent HTTP[6, 16] (P-HTTP) has been proposed to reduce the setup cost in TCP. TCP fast start [16] is a technique that utilizes prior connections' information to avoid slow start of TCP thus speed up short file downloading. Performance of various HTTP transportation protocols was analyzed in [7]. The authors presented analytical models of HTTP over TCP, Asynchronized Reliable UDP Protocol (ARDP), T-TCP, and P-TCP. The major performance metrics used in these work was transmission time of the whole page. As discussed in the previous sections, this measurement may not reflect the overall perception of end users.

Using UDP to transmit Web traffic has been explored in [5, 17]. A simulation study in [5] has revealed the performance improvement under various link conditions by using UDP to transfer Web pages. A implementation-based study in [17] proposed a hybrid TCP/UDP protocol that exploits the lightweightedness and caching friendliness of UDP under low lossy links and switches to TCP otherwise. The major concerns in these studies are how to exploit the lightweight UDP to expedite the delivery of HTML pages and the extended caching benefit due to UDP's statelessness. Whereas our work only utilizes UDP's out-of-order delivery to minimize the user perceivable silent time.

Mogul has proposed in an Internet draft [14] to eliminate the HOL blocking by adding request ID in pipelined HTTP/1.1 requests. Our scheme is applicable to both HTTP/1.0 and 1.1, at the granularity of page fragments versus requests in the draft. Parallel downloading can relieve the HOL blocking. But parallel connections to the same Web server incurs more bursty network traffic. In [18], the downloading process was conducted through multiple ge-

ographically separated servers. As a result, the traffic are distributed over multiple links. To realize the scheme, however, multiple servers should be deployed in different network domains.

Fragment-based dynamic page generation and caching have been studied in [3, 15]. Dynamic content caching based on page fragmentation was proposed in [3] that uses a graphical representation of Web pages to exploit data dependency relationship. WebGraph [15] is a component-based server processing scheme to distinguish diverse attributes of each component and take appropriate measures on them. Although this paper is also based on the component-frame architecture, the way that a Web page is partitioned into various component is based on the surrounding tags and the content length. And the goal of this paper is to expedite Web page transmission and rendering whereas the other works are mainly intended to facilitate dynamic page generation and caching.

5 Conclusion

The head-of-line blocking in the process of Web page delivery occurs due to packet delay, loss and reordering. TCP's in oder data delivery to browsers and the long tag distance in Web pages contribute to the HOL blocking problem and results in longer user perceivable silent time. We propose a UDP-based out-of-order capable Web page transmission protocol to solve the problem. This protocol splits the structure and data in separate data frames such that the rendering of HTML data is not constrained by their enclosing tags. As a result, data blocks can be rendered immediately after their arrival at the end users without having to waiting for the complete context information. The proposed scheme amortizes the silent time and facilitates partial content caching and non-interactive applications. An implementation-based experiment validated the feasibility and demonstrated significant performance improvement under various network conditions.

References

[1] J. Bennett and C. Partridge, "Packet Reordering is Not Pathological Network Behavior," IEEE/ACM Transactions on Networking, Vol 7, No. 6, December, 1999.

[2] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating User-Perceived Quality into Web Server Design," In Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands , May 2000.

[3] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed, "A Publishing System for Efficiently Creating Dynamic Web Content," IEEE INFOCOM 2000.

[4] H. Chen and P. Mohapatra, "Session-Based Overload Control for QoS-Aware Web Servers," IEEE INFOCOM 2002.

[5] I. Cidon, R. Rom, A. Gupta, and C. Schuba, "Hybrid TCP-UDP Transport for Web Traffic,", IEEE IPCCC 1999.

[6] J. Heidemann, "Performance Interactions Between P-HTTP and TCP Implementations," ACM Computer Communication Review, vol. 27(2), pp. 65–73, April 1997.

[7] J. Heidemann, K. Obraczka, and J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols, " IEEE/ACM Transactions on Networking, vol. 5, no. 5, pp. 616–630, Oct. 1997.

[8] ICONOCAST newsletter for August 17, 2000, http://www.iconocast.com/issue/20000817.html

[9] J. Nielsen, "Chapter 5, Usability Engineering." Morgan Kaufmann, 1994.

[10] J. Nielsen, "Designing Web Usability." New Riders Publishing; December 16, 1999.

[11] NIST NET, `http://snad.ncsl.nist.gov/itg/nistnet/`.

[12] MAWI Working Group Traffic Archive, `http://tracer.csl.sony.co.jp/mawi/`.

[13] B. A. Myers, "The Importance of Percent-Done Progress Indicators for Computer-Human Interfaces." Proc. ACM CHI'85 Conf. (San Francisco, CA, 14-18 April), 11-17.

[14] J. Mogul, "Support for Out-of-order Responses in HTTP." `http://search.ietf.org/internet-drafts/draft-mogul-http-ooo-00.txt`.

[15] P. Mohapatra and H. Chen, "WebGraph: A Framework for Managing and Improving Performance of Dynamic Web Content," Special Issue of Proxy Servers in the IEEE Journal of Selected Areas in Communications, 2002.

[16] V. Padmanabhan and R. Katz, "TCP Fast Start: a Technique for Speeding Up Web Transfers," IEEE Globecom'98 Internet MiniConference, November 1998.

[17] M. Rabinovich and H. Wang, "DHTTP: An Efficient and Cache-Friendly Transfer Protocol for Web Traffic." IEEE INFOCOM 2001.

[18] P. Rodriguez, A. Kirpal, and E. W. Biersack, "Parallel-Access for Mirror Sites in the Internet", IEEE INFOCOM 2000.