Analysis of HTTP Performance

Joe Touch, John Heidemann, and Katia Obraczka
Aug. 16, 1996
USC / Information Sciences Institute
{touch, johnh, katia}@isi.edu

USC/ISI Research Report 98-463 / Dec. 1998

Abstract [1]

We discuss the performance effects of using per-transaction TCP connections for HTTP access, and the proposed optimizations of avoiding per-transaction re-connection and TCP slow-start restart overheads. We analyze the performance penalties of the interaction of HTTP and TCP. Our observations indicate that the proposed optimizations do not substantially affect Web access for the vast majority of users, who typically see end-to-end latencies of 100-250 ms and use low bandwidth lines. Under these conditions, there are only 1-2 packets in transit between the client and server, and the optimizations reduce the overall transaction time by only 11%. Rates over 200 Kbps are required in order to achieve at least a 50% reduction in transaction time, resulting in a user-noticeable performance enhancement.

Note: *This document first appeared on the web in June 1996, and was revised to its current form in August 1996, as http://www.isi.edu/lsam/publications/http-perf/. This is an archive of that version, with updated references only.*

1: Introduction

There have been several recent discussions about the performance problems of HTTP over TCP. This Web page continues that discussion with a description of the performance benefits of the proposed approaches. We discuss the evolution of the HTTP protocol, and the potential for inefficiencies. We then present analysis of these inefficiencies in current Web systems. We have found that, except for users with Ethernet-speed end-to-end links between the client and server, the performance enhancements proposed in [16] and [9] have limited benefit. The proposed application-layer persistent connection mechanisms achieve only a 11-27% reduction in response time for low bandwidth access, whereas we consider a 50% reduction in response time the minimum for substantial user benefit (i.e., the transaction is twice as fast). We conclude that such mechanisms are of limited benefit, and observe that they may interfere with emerging Internet services.

2: HTTP

The HTTP protocol was originally developed to reduce the inefficiencies of the FTP protocol [14], [1]. The goal was fast request-response interaction without requiring state at the server. To see the performance advantage of HTTP over FTP, we can compare the process of file retrieval transactions in each protocol. Both protocols use TCP, a reliable, connection-oriented transport protocol [13].

In FTP, a client opens a TCP connection with the server for control (Figure 1). Once that connection is established, a request for a file is sent on that channel. The server then opens a separate TCP connection for the file transfer, and returns the file in that other connection. Each connection requires one round-trip time (RTT) to open. The request takes 1/2 a RTT to get to the server, and the response takes another 1/2 RTT to return, in addition to the transmission time of the file. The overall time required for an FTP transaction is:

1 RTT control-channel OPEN
0.5 RTT send request on control-channel
1 RTT file-channel OPEN
0.5 RTT file starts to arrive
on file-channel
Ftrans time to transmit the file
----
3 RTT + Ftrans
= time to get the first file in FTP

1. This work is supported by the Defense Advanced Research Projects Agency through FBI contract #J-FBI-95-185 entitled “Large-Scale Active Middleware”. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Department of the Army, the Defense Advanced Research Projects Agency, or the U.S. Government.

This is shown in Figure 1, below. The control channel interaction is shown in dashed lines (red), annotated on the left, and the file channel is shown in solid lines (blue), annotated on the right. The file transfer itself is in grey.

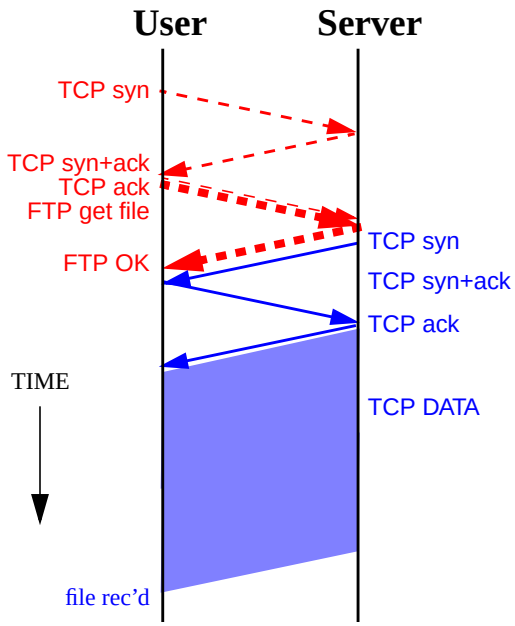

FIGURE 1. FTP File Transfer (first file)

Subsequent transactions to the same server may take less time, because the control channel is already open. A new TCP connection is required for each transaction in conventional use (block and compressed modes do not require this, but are not commonly used). The interaction is shown in Figure 2.

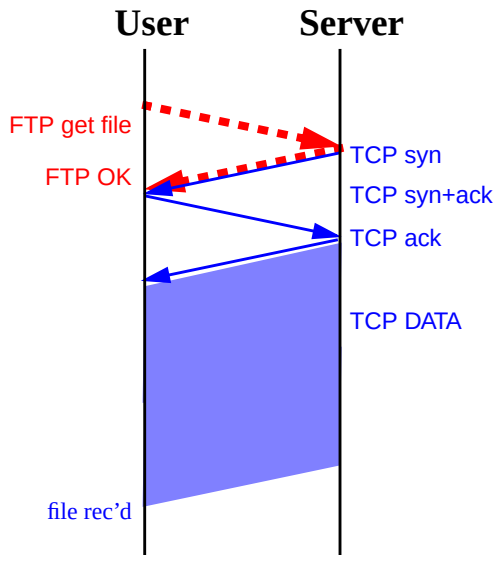

FIGURE 2. FTP File Transfer (subsequent files)

0.5 RTT send request on control-channel
1 RTT file-channel OPEN
0.5 RTT file starts to arrive on file-chan
Ftrans time to transmit the file
----
2 RTT + Ftrans
= time to get next files in FTP

HTTP uses a single TCP connection for the entire transaction, achieving FTP's best response time, even for the first file requested. Further, HTTP doesn't require the control-channel to be maintained at the server or client, so is stateless and simpler to implement. The transaction is also shown in Figure 3.

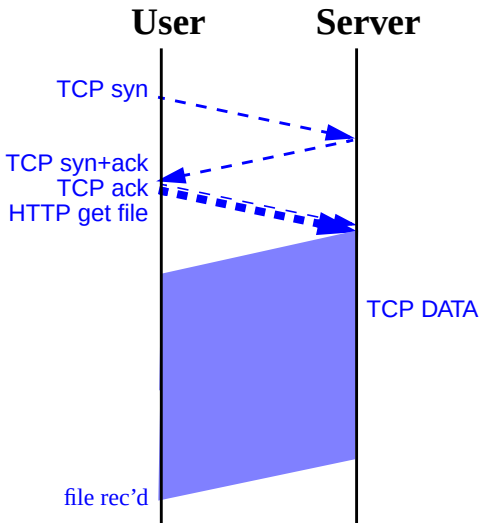

FIGURE 3. HTTP File Transfer

1 RTT channel OPEN
0.5 RTT send request
0.5 RTT file starts to arrive
Ftrans time to transmit the file
----
2 RTT + Ftrans
= time to get a file in HTTP

3: Potential Protocol Inefficiencies

There are inefficiencies in using HTTP over TCP. TCP establishes a connection prior to transferring any data, namely the request. TCP also includes a congestion avoidance mechanism [7]. In both cases, these mechanisms are restarted for each file request, possibly resulting in excessive overheads.

3.1: Connection Establishment

A minimal reliable transfer could occur with as little as one round-trip of overhead, plus the file transmission time, as shown in Figure 4.

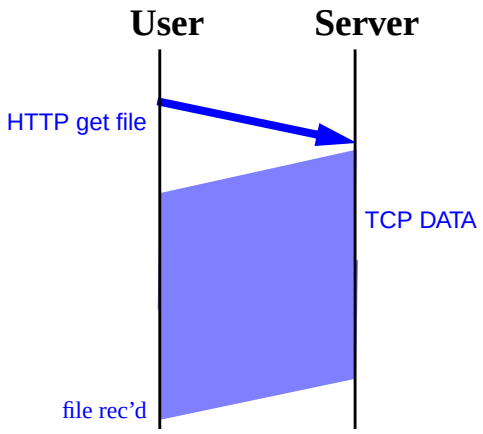

FIGURE 4. Optimal Transaction

1 RTT channel OPEN and send request, file starts to arrive
Ftrans time to transmit the file
----
1 RTT + Ftrans
= time to get a file optimally

TCP does not support the minimal transaction because the initial request cannot be delivered to the server until the connection has been established, which takes 1.5 RTTs, from the server's perspective (Figure 5) [13]. This is true even if the request is enclosed with the initial "SYN" OPEN packet; delivery of the request at the server is stalled until the third packet of the exchange arrives.

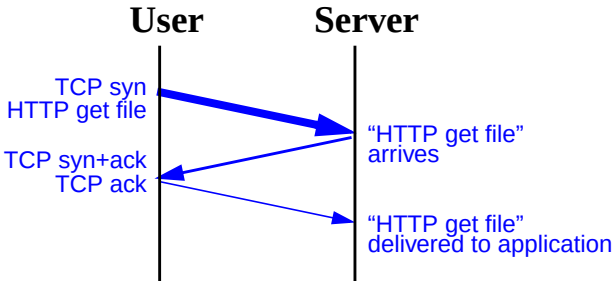

FIGURE 5. Delayed request deliver

3.2: Congestion Management

TCP also employs two congestion management mechanisms, one called "slow-start", and the other called "congestion avoidance" [7]. Slow-start prevents overwhelming the network when a connection begins by limiting the initial send window size, and allowing that window to grow in moderation to positive feedback. Congestion avoidance incorporates the negative feedback of packet loss, and modulates the send window as a result. Here were focus on best-case behavior, and so are primarily concerned with the slow-start mechanism.

In slow-start, when a connection opens, only one packet is sent until an ACK is received. For each ACK received, the number of packets that can be sent is increased by one. For each round-trip, the number of outstanding packets doubles, until a set of thresholds have been reached.

The packet size is negotiated. The default is 536 bytes in TCP, although many implementations round this down to 512. Hosts on Ethernets typically use 1460 for local connections. Where implemented, MTU discovery will allow Ethernet-sized MTUs on wide-area connections [10].

Slow-start occurs when a connection is initialized, when a packet is lost, or may occur when there is a significant idle period in the connection. The latter is described in [8], and implemented in 4.4BSD and derivatives, although it has not been adopted by earlier BSD-TCP users (for example, SunOS 4).

4: Alternative Protocol Mechanisms

There have been several proposals to address the potential inefficiencies of using HTTP over TCP. These include persistent-connection HTTP, Transaction TCP, and (recently) sharing TCP control blocks. These proposals address either connection or slow-start overheads, and in some cases, both issues.

4.1: Persistent HTTP

Persistent HTTP addresses both connection and slow-start overheads. There are several distinct proposals for P-HTTP, including [5], [15], and [9]. For the purposes of this discussion, we treat them as equivalent.

P-HTTP attempts to achieve optimal transaction time for sequences of transactions to the same server. The initial transaction occurs as in HTTP, but the connection is not closed. Subsequent requests occur without needing to re-open the connection.

In addition, P-HTTP attempts to avoid slow-start restart for each new transaction, again by using a single connection for a sequence of transactions. Unfortunately, sufficiently large gaps in the arrival of requests may cause a restart of slow-start anyway, due packet loss during the resulting packet burst when the transmission resumes, notably in the 4.4BSD-derived TCP implementations. The P-HTTP method is useful primarily for multiple adjacent

requests, as would occur on pages with embedded images, for example.

P-HTTP achieves this efficiency at the expense of application-layer complexity. Re-using a single connection requires application-layer multiplexing or can stall concurrent requests arbitrarily. Consider retrieving a large PostScript file, and issuing a small HTML file request during the transfer. The HTML response will either be stalled until the end of the PostScript file transmission, or the PostScript file will be segmented. The server cannot know whether this segmentation is required or not when it started to send the PostScript file. MIME-style headers are in-line and not encoded via "escapes"; only the specified length is used to determine when to parse the next header. As a result, the inefficiency of application-layer segmentation and reassembly occurs for every transaction. Finally, application-level multiplexing interferes with emerging Integrated Services multiplexing in the kernel, for Type-of-Service and Quality-of-Service mechanisms [4].

4.2: Transaction TCP

Transaction TCP (T/TCP) provides transaction-oriented service over TCP via extensions to the TCP protocol [2], [3]. T/TCP uses cached per-host state to avoid the delayed delivery of data carried with an OPEN, as discussed earlier. TCP delays that data to avoid delivery to the wrong connection, especially in cases of aborted connections. T/TCP uses cached values of extended state to avoid such errors, and permits early delivery before the third packet in the exchange.

In addition, T/TCP caches other TCP protocol control block parameters, such as round-trip time measures, to avoid inefficiencies with reconnecting to the same host. Reusing slow-start information, which would avoid slow-start restart, is discussed briefly in the T/TCP specification.

4.3: Shared TCP Control Blocks

Shared TCBs (S-TCB) augment the TCB state-sharing mechanism of T/TCP and show how to aggregate parameters such as window size across sets of concurrent connections [17]. T/TCP state caching is aimed predominantly at serial connection state reuse, whereas S-TCBs address both serial and concurrent shared state reuse.

S-TCBs optimize only the inefficiency of the slow-start restart component of HTTP over TCP. Also described in the S-TCB memo are the effects of application-layer multiplexing, and ways in which kernel-based multilevel feedback queuing, as in Integrated Services, would be adversely affected.

When S-TCB and T/TCP are coupled, they provide similar efficiency to P-HTTP, but at the kernel-level rather than requiring application-layer multiplexing.

5: Prior Analyses

Earlier analyses have claimed significant performance problems with HTTP over TCP [16], [11], [12]. Both Spero's and Mogul's analyses focused on client/server interactions in well-connected (1 Mbps) hosts [16], [12]. Their conclusions do not apply to the vast majority of web accesses, which are for small files over modem and ISDN links.

Moskowitz described performance problems at the server, where buffering limitations in the operating systems affected transaction performance. The claim is that the server runs out of buffers to create new TCP connections; the purported evidence is the "Host Connected, Waiting Reply" message. This message is emitted after the TCP connection is established, which is in turn after the TCB control block is allocated. This message is possibly evidence of processing bottlenecks at the server after the connection is established, although it is counter-proof of the claimed lack of buffers.

We are currently looking at HTTP performance over several transport protocols, including TCP, p-HTTP, T/TCP, and UDP-based RPC protocols, over a wider variety of network conditions [6]. This paper will contain both a more detailed model of HTTP performance than presented here and validation of this model against real-world traffic.

6: Environment Characteristics

The degree of inefficiency of HTTP over TCP depends on the environment in which the Web operates. Here we describe some characteristics of that environment. These factors will be used in the evaluation of the performance inefficiencies of HTTP over TCP later.

6.1: Networks

Current network environments can be characterized by a small set of classes: remote (satellite), modem, ISDN, leased-line direct (T-1), and high-performance (ATM, fast). These classes are named for dominant factor in the path between client and server, shown in Table 1

The default TCP MSS is 536 bytes for data, although most current implementations round this down to 512 bytes. Fast links support larger MSSs, but TCP often ignores them and uses the default for connections where MTU discovery is not implemented.

Net	BW (bps)	MSS (bytes)	Latency (ms) LAN/MAN WAN
satellite	9K	512	250 500
modem	29K	512	150 250
ISDN	112K	1460/512	30 130
direct	1M	1460/512	2 100
fast	155M	8192/512	2 100

TABLE 1. Network properties

6.2: The Web

Current Web use can also be characterized by classes of file types accessed. Several studies have shown that the vast majority of Web accesses retrieve small files, on the order of 6 KB. We describe these classes as shown in Table 2.

Web	File size (bytes)	File Type
HTML	6K	ASCII text
Web page	6+2+2K	HTML and links (2 icons)
text	60K	ASCII text
icon	2K	small GIF (icon)
image	20K	large GIF (clickable map)
photo	200K	very large GIF (photo)

TABLE 2. Web page properties

Although this describes the characteristics of web pages in general, the majority of accesses are to 6 KB files, and that is the focus of the discussion. Other analysis at ISI shows that the “Web page” case has a greater potential for optimization than the HTML case, because it is composed of multiple files [6]. In particular, the aggregate of files denoted by a Web page can be retrieved in a single connection with a single aggregate ‘GET-ALL’ request, rather than even using persistent connections [12].

The effect of the optimizations (avoiding connection establishment and slow-start) depends on the network properties; for modem links it is 11%, for ISDN it is also low, around 27%. Optimization increases significantly for faster connections or for higher latency paths [6], which is similar to the results in [16] and [9].

7: Evaluation

We want to determine the potential for inefficiency in HTTP over TCP. For this purpose, we analyze the time required for an HTTP interaction, computing an upper-bound for both the per-transaction connection establishment and potential slow-start overheads, and compare that to the optimal time for transfer.

These analysis consider optimal performance of the system. We assume that the hosts are limited only by the network bandwidth, that server processing time is negligible, and that disk I/O and other bottlenecks are minimal. We consider the worst-case performance of HTTP over TCP, assuming no packet loss. This maximizes the benefit of persistent connections.

Even so, these optimizations benefit most Web users only minimally. When other factors, such as server processing, packet loss, etc., are included, the optimizations are even less noticeable.

The following section presents analysis for HTTP over TCP. A more complete analysis of HTTP over several transport protocols and P-HTTP is currently underway [6]. The formulae below are simplified versions originally developed there. The formulae in this paper provide an upper bound on performance, the formulae in [6] are more precise and include implementation-specific interactions.

7.1: Analysis

The following notation is used in the analysis:

R = RTT

bw = bandwidth

MSS = max. segment size (packet size)

K = number of packets in the file
= filesize / MSS

L = round trip time in packets,
i.e., length of the pipe
= number of packets to fill the pipe
= bw * R / MSS

M = max useful window size
(lower bound)
= min(L, K)

S = round trips stalled in slow-start, assuming no loss (upper bound)
(window starts at 2, see [6]**)**
= floor(log2(ceil(M/2)))

W = amount of wasted time
= (upper-bound on waste --
not all slow-start is wasted, though)
= slow-start + connection-setup
= R * S + R

F = min. file transmission time
= filesize / bw

Tmin = min. transaction time
= F + R

T = transaction time
= Tmin + W (discounting server processing time)

R, bw, MSS, K, and L are self-explanatory. M is the maximum useful window size for this file and network, bounded by the smaller of the packets in round-trip (L) and the packets in the file (K).

S is the number of round-trips stalled during the initial slow-start. The initial send window starts at 1 MSS, but in most BSD implementations it is increased to 2 when the TCP SYN (connect start) is ACK'd. The window for data transport effectively begins at 2, and doubles each round-trip. The transmission stalls each time this window is smaller than M. Each stall wastes at most one round-trip; actually, the entire round-trip time is not wasted, since the window was non-zero, but this is an upper-bound.

W is the amount of wasted time, the total of one round trip for connection establishment, and at most one round trip for each slow-start stall. We compare this number to the absolute minimum for a transaction, composed only of one round-trip for the request exchange and the file transmission time. We assume here that the request transmission time is negligible compared to the file transmission time; typically requests are less than 100 bytes. We also ignore header overheads.

The amount of wasted time becomes noticeable to the user when it is the same as the minimum transaction time, or larger. At that point, removing re-connection and slow-start restart overheads will halve the time of access. This assumes no network loss.

Tmin + P <= W

(under best conditions, assume P goes to zero)

Tmin <= W

So we can plot the ratio of time wasted to file transmission, as an upper-bound on the optimization possible. This ignores processing time and other impediments at the server, as mentioned earlier. We also count the entire round-trip of each slow-start exchange as wasted, which is not strictly true. Up to one RTT-worth of data is sent during this exchange, at most; by ignoring this, we achieve a further upper-bound.

There are two ratios we can compare. The first is percent wasted time, the ratio of waste to optimal file transaction time. The second, more intuitive notion is percent of possible reduction, the ratio of waste to transaction including waste. The first is more meaningful experimentally, because the waste varies independently of the optimal transaction time, and so the ratio varies in the numerator only. The second is easier to intuit; a 10-second transaction with 50% possible reduction optimizes to 5 seconds, whereas in the first equation this would be a 1:1 ratio. We will show both the ratio and the percent possible reduction.

$$\frac{W}{Tmin} = \text{ratio wasted time}$$

$$\frac{W}{Tmin + W} = \%\ \text{of possible reduction}$$

7.2: Some common cases

We plotted the ratio of waste to useful time on a contour plot. For a given network RTT, we want to see what bandwidth is required for the proposed optimizations to reduce the overall transaction by half, i.e., where the waste is the same as the useful time.

The graph is fixed for a constant filesize of 6 KB. We consider bandwidths from 10 Kbps - 1 Mbps and latencies from 0.01-1 seconds. Typical latencies are 70 ms for end-to-end latency for average Web surfing in the USA, with 30-150 ms of additional latency for modem or ISDN links. We therefore consider 250 ms total latency for modem links and 100 ms total latency for other types of directly-connected networks. Satellite network latencies are higher, but not considered below.

Shown in Figure 6 are contour lines where the waste to useful time is 0.25:1, 0.50:1, 2:1, and 3:1. I.e., for 1:1, removing the overhead halves the effective transaction time. The shaded area shows where this 2x speedup (or greater) applies. For this graph, we consider Internet interactions, so that the MSS is 512 bytes.

Current modem links, at 28.8 Kbps and 250 ms total round-trip latency, have only 11% waste for a 6 Kbyte file transfer (A), a ratio of 1.13:1. Waste of 50%, i.e., a ratio of 1:1, is achieved around 100 Kbps at 250 ms latency (B).

ISDN (112 Kbps) at 100 ms latency has 27% waste (C), a ratio of 0.37:1. At 100 ms latency, 260 Kbps end-to-end links are the minimum required to approach the waste ratio of 1:1, i.e., the 2x speedup sought (D).

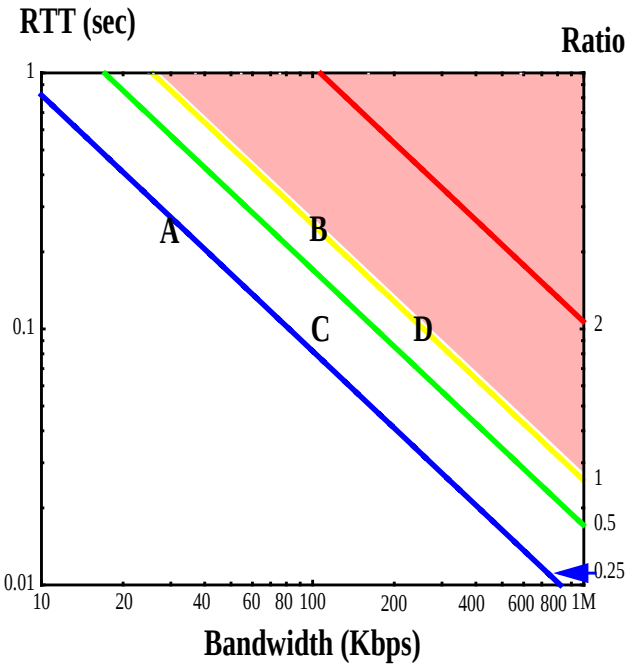

FIGURE 6. effect of optimizations (Internet MSS) Contour plot of (wasted time/useful time)

7.3: MODEM (Internet MSS)

The following equations govern the benefit for modem links. As indicated, the potential benefit is an 11% reduction in wasted time.

R = 0.250 s
bw = 28,800 bps
MSS = 512 Bytes = 4096 bits
(filesize = 6 KB)
K = 12 packets
L = 1.76
M = min(1.76, 12) = 1.76 packets
S = floor(log2(ceil(1.76/2))) = 0 rtts
W = 0.250 s

F = 1.71 s

W/Tmin = waste ratio is 0.13:1

W/(Tmin+W) = % poss. benefit is 11%

7.4: ISDN (Internet MSS)

The following equations govern the benefit for ISDN links. These equations assume Internet MSS (512 bytes), end-to-end. As indicated, the potential benefit is a 27% reduction in wasted time.

R = 0.100 s
bw = 112,000 bps
MSS = 512 Bytes = 4096 bits
K = 12 packets
L = 2.73 packets
M = min(2.73, 12) = 2.73 packets
S = floor(log2(ceil(2.73/2))) = 1 rtt
W = 0.200 s
F = 0.44 s

W/Tmin = 0.37:1 waste ratio

W/(Tmin+W) = 27% possible benefit

We re-evaluated the graph for the case where MTU discovery is implemented, and packets contain the Ethernet-MSS (1460 bytes), as shown in Figure 7. In this case, the results of the optimizations are different. End-to-end rates of 80 Kbps are required at 250 ms latency (E), and 200 Kbps is required for 100 ms latency (F). ISDN at 100 ms gains only 16% from the optimizations.

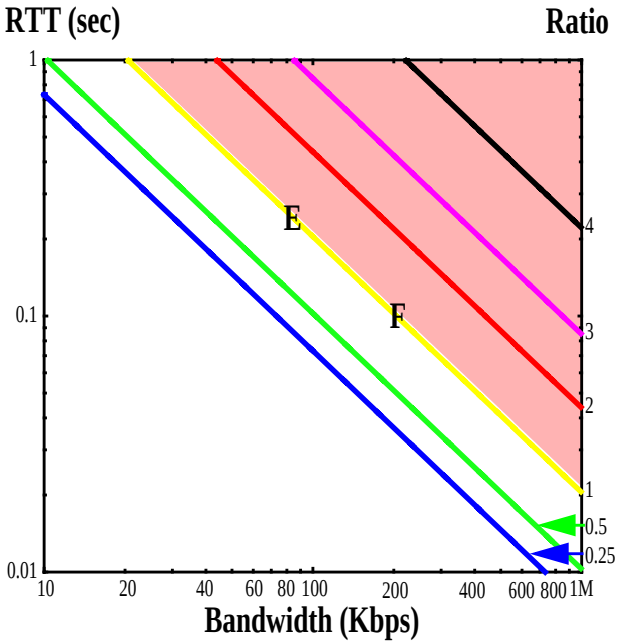

FIGURE 7. Effect of optimizations (Ethernet MSS) Contour plot of (wasted time/useful time)

7.5: ISDN (Ethernet to ISDN)

The following equations govern the benefit for ISDN links connected to local Ethernets, supporting and end-to-end MSS of 1460 bytes. As indicated, the potential benefit is 16%. Note that this is lower than the Internet MSS version of an ISDN line, under otherwise identical conditions.

R = 0.100 s
bw = 112,000 bps
MSS = 1460 Bytes = 11680 bits
K = 4.21 packets
L = 0.96 packets
M = min(0.96, 4.21) = 0.96 packets
S = floor(log2(ceil(0.96/2))) = 0 rtt
W = 0.100 s
F = 0.44 s

W/Tmin = 0.19:1 waste ratio

W/(Tmin+W) = 16% poss. improvement

These observations indicate that avoiding connection establishment and slow-start does not benefit current Web access for the vast majority of users. Most users see end-to-end latencies of about 250 ms and use modem lines. At these rates, the optimizations reduce the overall transaction time by 16%. Rates over 200 Kbps are required to provide user-noticeable performance.

8: Conclusions

Our analysis indicates that the persistent connection optimizations do not substantially affect Web access for the vast majority of users. Most users see end-to-end latencies of about 250 ms and use modem lines. At these rates, the optimizations reduce the overall transaction time by 11%. Bandwidths over 200 Kbps are required to provide user-noticeable performance improvements.

To be noticeable, the optimizations require network and file characteristics that are not true for most users. Connection establishment optimizations require that the file is as small as the round-trip bandwidth-delay product, or smaller. Slow-start optimizations require that there are a large number of packets in the round-trip, and that the overall number of packets is a small number of round-trips' worth. Neither of these assumptions hold for users over modem or ISDN lines accessing the vast majority of Web files.

In such cases, only 1-2 packets are typically in transit in the round-trip, negating the effects of slow-start optimizations. Typically, files are over 10x larger than the round-trip bandwidth-delay product, negating the effects of connection establishment optimizations.

In the future, bandwidths are sure to increase. Packet sizes are also likely to increase, e.g., to 9 Kbytes for ATM, and MSS discovery should be more widely available. File sizes may increase as well. Given all three of these advances, it is not easy to predict the overall effect. This is discussed in further detail in ongoing work [6].

9: Acknowledgments

We would like to thank the members of ISI's HPCC Division, especially Ted Faber, for their assistance with this document. This document was the result of discussions on the http-ng and web-talk mailing lists, and we also thank the members of those lists for their feedback.

10: References

[1] Berners-Lee, T.J, R. Cailliau and J.-F. Groff, The World-Wide Web, Computer Networks and ISDN Systems 25 (1992) 454-459. Noth-Holland.

[2] Braden, R., "Extending TCP for Transactions -- Concepts," RFC-1379, USC/ISI, November 1992.

[3] Braden, R., "T/TCP -- TCP Extensions for Transactions: Functional Specification," RFC-1644, USC/ISI, July 1994.

[4] Clark, D., Shenker, S., and Zhang, L., "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," Sigcomm '92, pp. 14-26.

[5] Fielding, et al., "Hypertext Transfer Protocol - HTTP/1.1," (working draft), June 7, 1996.

NOTE: This became RFC 2068 in Jan. 1997.

[6] Heidemann, J., Obraczka, K., and Touch, J., "Analysis of HTTP Transport Protocols," (in progress).

NOTE: This was published with the revised title "Modeling the Performance of HTTP Over Several Transport Protocols," ACM/IEEE Transactions on Networking 5(5), 616-630, October, 1997.

[7] Jacobson, V., "Congestion Avoidance and Control," ACM Sigcomm '88, August 1988.

[8] Jacobson, V. and Karels, M., "Congestion Avoidance and Control,"

NOTE: This is a revised version of [7], which adds Karels as co-author and includes an additional appendix. The revised version has not been published, but is available at ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[9] Mogul, J., "The Case for Persistent-Connection HTTP," ACM Sigcomm '95, August 1995, pp. 299-313. A longer, more comprehensive version of this paper is available on line at Digital Equipment Corporation Western Research Laboratory Research Report 95/4, May, 1995.

[10] Mogul, J., and S. Deering, S., "Path MTU Discovery", RFC-1191, DECWRL, Stanford University, November 1990.

[11] Moskowitz, R., "Why in the World is the Web So Slow?" Network Computing, March 15, 1996, pp. 22-24.

[12] Padmanabhan, V., and Mogul, J., “Improving HTTP Latency,” Proc. of the Second International WWW Conference, Oct. 1994.

[13] Postel, J., “Transmission Control Protocol,” RFC-793 / STD-007, USC/ISI, September 1981.

[14] Postel, J., and Reynolds, J., “File Transfer Protocol (FTP),” RFC-959 / STD-009, USC/ISI, October 1985.

[15] Spero, S., “Progress on HTTP-NG,” (URL)

NOTE: This document was never archivally published. The original URL was http://www.w3.org/pub/WWW/Protocols/HTTP-NG/http-ng-status.html

[16] Spero, S., “Analysis of HTTP Performance Problems,” (URL)

NOTE: This document was never archivally published. The original URL was http://sunsite.unc.edu/mdma-release/http-prob.html

[17] Touch, J., “TCP Control Block Interdependence,” (work in progress), USC/ISI, June 1996.

NOTE: This became RFC-2140 in April 1997.