Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments*

Anja Feldmann Ramón Cáceres Fred Douglis Gideon Glass† Michael Rabinovich
AT&T Labs–Research, Florham Park, NJ, USA

Abstract

Much work on the performance of Web proxy caching has focused on high-level metrics such as hit rates, but has ignored low-level details such as *cookies*, aborted connections, and persistent connections between clients and proxies as well as between proxies and servers. These details have a strong impact on performance, particularly in heterogeneous bandwidth environments where network speeds between clients and proxies are significantly different than speeds between proxies and servers.

We evaluate through detailed simulations the latency and bandwidth effects of Web proxy caching in such environments. We drive our simulations with packet traces from two scenarios: clients connected through slow dialup modems to a commercial ISP, and clients on a fast LAN in an industrial research lab. We present three main results. One, caching persistent connections at the proxy can improve latency much more than simply caching Web data. Two, aborted connections can waste more bandwidth than that saved by caching data. Three, "cookies" can dramatically reduce hit rates by making many documents effectively uncacheable.

1 Introduction

The continued growth of the World Wide Web (WWW) motivates techniques to improve its performance. One popular technique is *proxy caching*, in which one or more computers act as a cache of documents for a set of WWW clients. These clients are configured to send HyperText Transport Protocol (HTTP) requests to the proxy. If possible, the proxy serves the requests from its cache. Otherwise, the proxy forwards the request to the *content provider*, that is, to the server containing the source copy of the requested data.

WWW proxy caching attempts to improve performance in three ways. First, caching attempts to reduce the user-perceived latency associated with obtaining Web documents. Latency can be reduced because the proxy cache is much closer to the client than the content provider. Second, caching attempts to lower the network traffic from the Web servers. Network load can be lowered because documents that are served from the cache typically traverse less of the network than when they are served by the content provider. Finally,

*A summary of preliminary results of this study was reported at the ACM SIGMETRICS Workshop on Internet Server Performance and is available at http://www.cs.wisc.edu/ cao/WISP98/final-versions/anja.ps.

†Current affiliation: Cobalt Microserver, Inc., Mountain View, CA, USA

proxy caching can reduce the service demands on content providers since cache hits need not involve the content provider. It also may lower transit costs for access providers.

Many attempts to model the benefits of proxy caching have looked only at the high-level details. For instance, one might consider a stream of HTTP request-response pairs, knowing the URL requested and the size of each response, and compute a *byte hit ratio*: the fraction of the number of bytes served by the cache divided by the total number of bytes sent to its clients. Hit rates estimate the reduction in bandwidth requirements between the proxy and the content providers; however, they do not necessarily accurately reflect the latency perceived by the end user, and in some cases they do not even reflect actual bandwidth savings. (The work of Kroeger, et al. [10], is a notable exception in its attention to end-user latency; however, it is directed at a different computing environment.)

This paper considers performance implications of factors that cannot be modeled at such high level. Notably, we evaluate the performance effects of sometimes drastically different bandwidth between clients and the proxy and between the proxy and the Internet. Other factors that we consider include aborted transfers, HTTP headers that may affect cacheability of resources, and persistent TCP connections.

We have developed a web proxy cache simulator that provides the level of detail necessary to model these factors. It uses workloads from two different environments. One trace was collected in a heterogeneous bandwidth production environment, AT&T WorldNet. The second trace was collected at AT&T Labs–Research; a well-networked research lab. The simulator considers HTTP operations at the level of individual TCP packets, including the slow-start phase at the beginning of each connection. This simulation includes data in transit, demonstrating the effects of connection aborts. It uses all relevant HTTP request and response headers, so information that affects cacheability, such as the presence of cookies, is included. Finally, it uses a combination of measured and parameterizable timing characteristics, allowing us to simulate a user community over a relatively slow modem pool, or a faster local area network.

Our study found that the bandwidth mismatch between the clients and the proxy and the proxy and the Internet can negate any bandwidth savings one might hope to reap from a proxy cache. In fact, the bandwidth consumption may *increase*, due to partially transferred data for aborted requests. We also observed that in such heterogeneous networks, the benefits of using persistent connections between clients and the proxy can outweigh the benefits of caching documents. We therefore argue for a dual role of the proxy as a *data cache* and a *connection cache*. While an infinite-size data cache reduces average user-observed latency in the low-bandwidth environment by only 8%, just caching connections alone results in up to 25% improvement in the average latency. Combined data and connection caching produces up to 28% latency reduction. In the high-bandwidth environment, data caching improves mean latency by 38%, connection caching improves it by 35%, but the combination of the two improves it by 65%.

Another finding is a surprisingly high fraction of responses with a *cookie* header. Since cookies are methods of customizing resources on a per-user basis, it is inappropriate to cache HTTP/1.0[1] resources that have cookies in them even though they might just be inlined images. In our client trace, we found roughly 30% of requests had cookies, which dramatically limits the number of requests that can possibly be satisfied from a proxy cache. Moreover, we have evidence that the use of cookies is increasing.

The rest of this paper is organized as follows. Section 2 describes the tracing environment, and Section 3 the simulator. Section 4 presents our results. Section 5 discusses related work and Section 6 concludes.

2 Tracing Environment

Our study is based on two traces, reflecting two common network environments for Internet access. One, referred to as the *modem trace*, contains accesses by subscribers to a commercial ISP, connected through slow modem lines. The other, called the *research trace*, contains accesses from a research community connected to the Internet via a high-speed link.

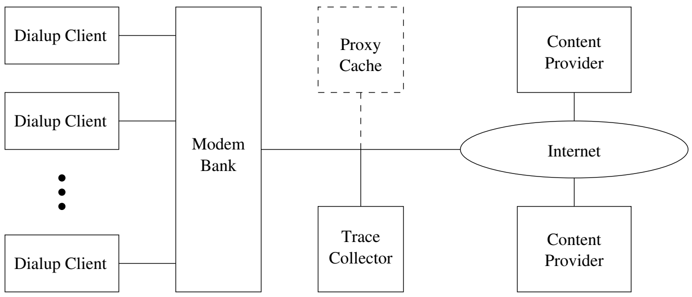


Figure 1: *Trace gathering and proxy caching environment for AT&T WorldNet.*

The modem trace was collected from a FDDI ring that connects an AT&T WorldNet modem bank to the rest of the Internet. In this environment, the proxy caches would reside in the ISP's facilities near such modem pools. Figure 1 depicts this scenario. The modem bank in question contains roughly 450 modems connected to two terminal servers, and is shared by approximately 18,000 dialup users. The servers terminate Point-to-Point Protocol (PPP) connections and route Internet Protocol (IP) traffic between the users and the rest of the Internet.

[1] In HTTP/1.1, caching and cookies are decoupled, and a server is responsible for explicitly disabling caching when appropriate.

We collected raw packet traces on a dedicated 500-MHz Alpha workstation attached to the FDDI ring. We ensured that the monitoring was purely passive by configuring the workstation's FDDI controller so that it could receive but not send packets, and by not assigning an IP address to the FDDI interface. We controlled the workstation by connecting to it over an internal network that does not carry user traffic. The traces were anonymized as soon as they came off the FDDI link, before writing any packet headers to stable storage.

We traced all dialup traffic on the FDDI ring for 12 days in mid-August 1997. During this time, 17,964 different users initiated 154,260 different dialup sessions, with a maximum of 421 simultaneously active sessions. Our tracing infrastructure handled more than 150 million packets a day with less than 0.3% packet loss. We obtained a second trace of all dialup traffic on the FDDI ring for 6 days in mid-July, 1998.

We processed the raw packet stream on the fly to obtain our final trace. The resulting trace contains all relevant information and is much more compact than the raw packet data. The trace records the following information for TCP conversations in which one port number is the default HTTP port (80):

- TCP events: Timestamps, sequence numbers, and acknowledgments for all packets with SYN, FIN, or RST bits set.
- HTTP events: Timestamps for HTTP requests from clients and HTTP responses from servers, and timestamps for the packets containing the first and the last portion of the data in each direction.
- HTTP headers: Complete HTTP headers for both requests and responses.
- Byte counts: Count of bytes sent in either direction for HTTP header and body, if present.

This information is sufficient to determine how much time is taken by various components of the observed HTTP conversations. These traces are significantly more detailed than traces that other researchers have made available [8, 10]. For example, publicly available traces lack timestamps for TCP events such as connection creation (SYN packets).

For the research trace, we recorded accesses from the AT&T Labs–Research community. The tracing environment for collecting the research trace is shown in Figure 2. The trace was gathered from an Ethernet segment adjacent to the serial line that connects AT&T Labs–Research to the rest of the Internet. We used the same trace collection software as in the modem case. At the time of the trace collection this serial line was a T1 link (1.5 Mbit/s). The raw packet traces were collected on a dedicated 133-MHz Intel Pentium PC running Linux. Again we isolated the tracing from the traced network.

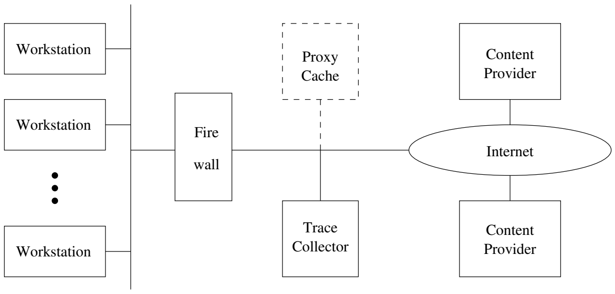


Figure 2: *Trace gathering and proxy caching environment for AT&T Labs–Research.*

We traced the high speed access clients for 11 days in mid-February 1997. During this time, more than 3,000 client machines accessed more than 23,000 external Web servers. A subset of this trace was used to study overheads in flow-switched networks[6].

3 Simulator

Our web proxy simulator, named *PROXIM*, simulates a proxy cache using the HTTP trace (described in Section 2) as input. *PROXIM* can be used to simulate the three different scenarios shown in Figure 3,

a. using a proxy,

b. without a proxy, where the bandwidth to the clients is the bottleneck, and

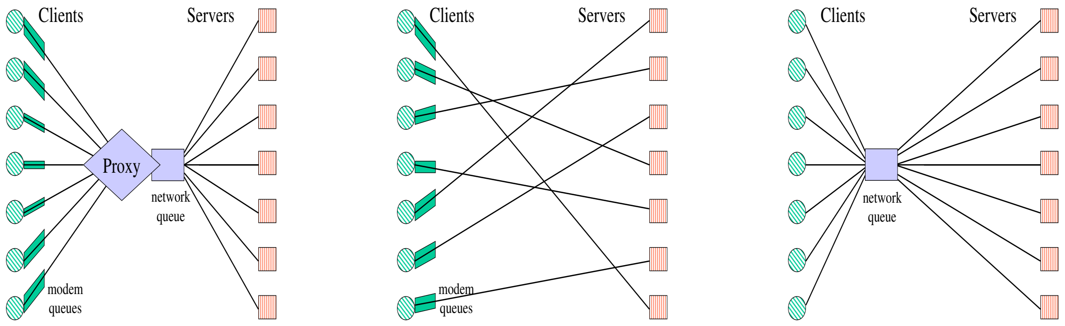
c. without a proxy, where the bandwidth on the network connecting the clients to the Internet is the bottleneck.

To assess the performance impact of proxy caching in a given environment, we compare the with-proxy to the no-proxy simulation results. In addition, the no-proxy simulation results are used to validate the simulator against the original measured numbers.

The subsections 3.1 through 3.4 describe various aspects of *PROXIM*'s functionality. Subsection 3.5 validates the simulator against the original trace.

3.1 Simulated Cache

PROXIM simulates a document cache managed by LRU replacement. However, in all of our experiments we configured the cache size to be sufficiently large to avoid having to evict any documents. This gives potentially optimistic results for the proxy, but the storage consumed by all cacheable documents in our trace



(a) Using a proxy.

(b) No-proxy; client bandwidth is the bottleneck.

(c) No-proxy; bandwidth to the Internet is the bottleneck.

Figure 3: *Proxy caching environment.*

is on the order of 40GB, not an unreasonable amount of disk space. To account for the overhead introduced by the proxy a configurable cache miss time overhead and a configurable cache hit time overhead are added to the request service time.

3.2 Network Connections

As mentioned in the introduction, it is important to handle the interactions between HTTP and TCP, particularly persistent HTTP [7, 12, 15]. In *PROXIM*, each simulated client maintains zero or more open connections to the proxy. An *idle* connection is one in which no HTTP request is currently active. When a persistent HTTP request is generated and an idle connection exists for the client in question, an idle connection is chosen to service the request. By default, requests that were marked non-persistent in the original trace (due to client of server header fields) are treated as non-persistent by *PROXIM* and result in a new connection. If a client generates a request and all of its extant proxy connections are active servicing other requests—or if the client has no extant connections—a new TCP connection is created by the client to service the present request. Client-to-proxy connections that are idle for more than a default of 3 minutes are closed by the proxy.

PROXIM handles proxy to content-provider connections similarly to the way it handles proxy to client connections. However, since proxy administrators have no influence over arbitrary content-providers, we time out persistent server connections after a somewhat arbitrary default of 30 seconds of idle time.

PROXIM assumes that a client requests a Web page at the time that the original client sent out the SYN packet (TCP connection request). Therefore, if the client can reuse a persistent connection, then the proxy learns about an HTTP request at the time at which we recorded the SYN packet. If the client is forced to open a new connection to the proxy, the proxy sees the request at one client-to-proxy round-trip time after the original SYN packet was observed.

3.3 Document Transfer

PROXIM simulates the transfer of documents to clients over a connection by scheduling individual packets to arrive at the client, the proxy, or the server. The scheduling of packets takes into account TCP slow-start at the start of a connection, but does not consider packet losses and the additional slow-start periods those would entail. By default, the proxy and servers send 1500-byte packets (the most common packet size besides 40-byte ACK packets).

Packets destined for the clients are enqueued in a per-client queue for scenarios (a) and (b) and in a network queue for scenario (c) of Figure 3. Packets destined for the proxy are enqueued in a network queue. Each of these queues drains at a configurable rate. The default for the modem trace is a server-to-proxy rate of 45 Mbps while the default rate for client queues is 21 Kbps. The first was chosen since a T3 (45 Mbps) constitutes a good connectivity to the Internet and the later was chosen since most modems connection speeds at this modem pool are in the order of 24 Kbps to 28 Kbps and we observed that they can sustain a throughput around 21 Kbps. The default for the research trace is a server-to-proxy rate of 1.5 Mbps and a network queue of 1.5 Mbps. These values were chosen since a T1 (1.5 Mbps) constitutes good connectivity to the Internet for a corporate environment.

This scheduling of packets requires a round-trip time estimate for the network link on which packets are scheduled. We use a constant round-trip time estimate for each type of connection, which for client-to-proxy connections in the modem trace is the modem delay (default 250 ms). For proxy-to-server connections the round-trip time is derived from either the difference between the SYN and SYN-ACK timestamps or the REQUEST and RESPONSE timestamps. (We assume that the proxy is located at the location of the packet sniffer, where the trace was collected.) The estimate for the client-to-server connection is done in a similar manner except that we add the modem delay of 250 ms in the case of the modem trace. For the research trace we assume a default round-trip delay of 2.5 ms between the clients and the proxy.

3.4 Latency Calculations

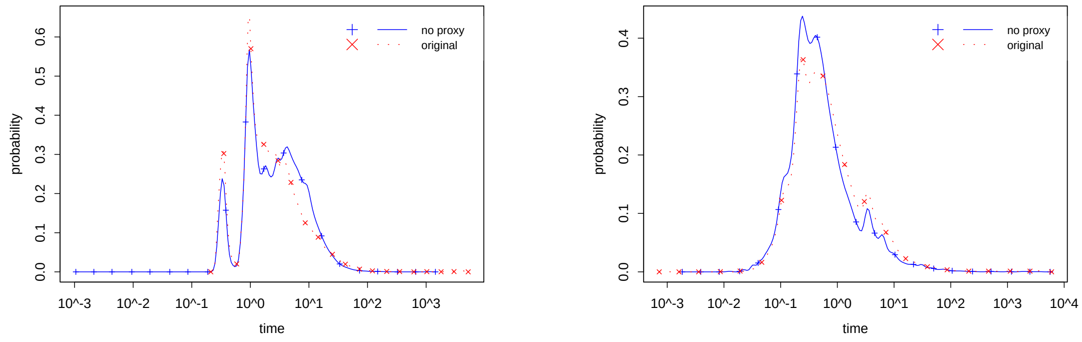
PROXIM simulates the overall latency to retrieve a document by breaking the latency overhead into connection setup time, HTTP request-response time, and document transfer time. Connection setup costs are avoided when a persistent connection is reused. In the no-proxy case (in which we simulate direct connections between clients and content-providers) we use the SYN timestamps in the trace to determine the connection setup time. In the with-proxy case, when a client establishes a connection with the proxy, or the proxy does so with a content-provider, we use the relevant subset of the SYN packet timestamps. Similarly, request-response latency is calculated using the appropriate portion of the HTTP request/reply latencies in the trace: the full latency is used if the proxy has to connect to the content provider; otherwise the request/response latency is assumed to be the configured client/proxy round-trip time. We modeled the time clients take to retrieve data from the proxy by using the queues mentioned above, which drain at modem bandwidth. We break documents into packets and observe the time difference between the start of the first downloaded packet and the dequeuing of the last packet.

3.5 Simulator Validation

To validate the simulator, we compare our results for the no-proxy simulations where either the client bandwidth is the bottleneck (modem environment) or where the bandwidth to the Internet is the bottleneck (research lab environment) and compared simulated latencies against the original latencies from the trace. Figure 4 shows the probability density of latency distribution in both cases. The simulation curve matches the general shape of the measured curve.

In particular, the median latencies for the data transfers in the original trace in the modem environment is 0.44 seconds vs. 0.42 seconds for the no proxy simulation. The median total latency for the no-proxy simulation is 2.4 seconds vs. 1.8 seconds for the original trace. This difference is due to the overestimation of message round-trip time in the simulator, which is calculated as the minimum of SYN/SYN-ACK and HTTP request/response delays, both of which require extra server processing. On the other hand, the mean latency of the original trace is larger than the mean latency of the no-proxy simulation since the no-proxy simulation does not account for for network congestion or content server overload, which might limit the data deposit rate into the modem queues. Note that this simulation is assuming that the client bandwidth is the bottleneck.

For the research environment, most of the clients are connected via a 10Mbit/s Ethernet. Therefore



(a) Modem environment

(b) Research environment

Figure 4: *Simulation validation: total latency.*

we follow the scenario of Figure 3 (c). In this case we choose the bandwidth of the network queue to be 1.5 Mbps (T1 speed) and assume that the network queue is drained at 6 Mbps (a reasonable assumption given a 10 Mbps Ethernet). The median latencies for the original trace in the research environment is 0.42 seconds vs. 0.54 seconds for the no-proxy simulation. Again the mean latency of the original trace exceeds the mean latency of the no-proxy simulation. Overall, both simulations reasonably represent the original trace.

4 Performance Effects of the Proxy

This section describes some insights into performance effects of a proxy within an ISP either serving a set of modems or a set of clients connected via a high speed connection, gained by considering low-level details that are often overlooked in similar studies.

4.1 Hit Ratio

Most performance studies of proxies to date have concentrated on hit ratios. We contend this is only a secondary measure, useful only insofar as it affects the performance metrics that users and network administrators ultimately care about, such as bandwidth savings and latency reduction.

Still, even when looking at just hit ratio, considering low-level details provides some interesting results. Unlike existing studies, our trace captures all HTTP headers of requests and responses, so we were able to emulate the full behavior of proxies with respect to cacheability of resources. There are a variety of reasons for a document to be uncacheable:

- Dynamically generated content, usually identified by the presence of “cgi-bin” or “?” in the URL.
- Explicit cache control through the use of Cache-Control and Expires header fields [7].
- A *cookie* is present: by using a Set-Cookie HTTP response header line, the content-provider has requested that the client send the content-provider a “cookie” with each subsequent request. The cookie is an arbitrary sequence of bytes, typically used to identify the client. The client sends the cookie by means of a Cookie header line. Cookies are often used for authorization and/or personalization of Web pages. Although the Cookie and Set-Cookie header fields are not (yet) officially part of HTTP and thus have no inherent caching-related requirements, current proxy caching software should not cache the results of HTTP 1.0 requests with Cookie header lines. By default, proxy software (e.g., Squid [18]) does not cache results of requests with Cookie header lines.
- An authorization field is present in the request: this would require proxies to cache content on a per-client basis, or unauthorized users might see content meant for another (authorized) user.

The following table lists the number of HTTP requests that were uncacheable for the various reasons. The counts are non-exclusive, but the last line of the table counts the union of the individual causes.

Reason	Count modem trace	% modem trace	Count research trace	% research trace
Cookie present	3,236,869	30.20	196,071	18.83
URL had '?'	1,055,470	9.85	77,692	7.46
Client Cache-Control	800,574	7.47	149,720	14.38
URL had 'cgi-bin'	580,451	5.42	48,753	4.68
Not GET or HEAD method	216,073	2.02	19,336	1.86
Authorization present	180,728	1.69	11,505	1.10
Server Cache-Control	177,404	1.65	11,412	1.10
All uncacheable documents	4,615,661	43.06	391,064	37.55

Surprisingly, the ISP trace showed that over 30% of all requests had a cookie, which had a dramatic effect on the hit ratio. If the proxy ignored cookies in the ISP, the hit ratio would have been 54.5%, similar to hit rates observed in previous studies [8, 10]. By taking cookies into account, we observe a hit ratio of just 35.2%, even when the cache size is large enough to avoid any purges. The byte hit ratio similarly decreases from 40.9% to 30.42%. In fact, we have preliminary evidence that the use of cookies is increasing. In a six-day trace we collected in June 1998, the fraction of requests with cookies grew to 34.5%.

For the somewhat older research trace, the impact of cookies is not quite as dramatic. The hit rate decreases from 33.9% to 27.0% if the proxy considers responses with cookies uncacheable; the byte hit rate decreases from 28.1% to 23.3%. The fact that the hit rate is much lower for the research trace than for the modem trace is likely to be correlated with the lower request rate and the difference in browsing behavior for researchers and consumers.

The significant increase of hit ratios produced by caching documents with cookies supports the importance of techniques aimed at enabling caching of such of documents. These techniques include approaches based on delta encoding [1, 9, 14] and client-side HTML macro-preprocessing [4]. The extent of the benefits of these techniques would depend on the amount of document customization based on the request cookies.

Another technique used by some proxy vendors is to ignore a cookie if the request appears to be for an image (identified by a suffix in the URL) [3]. This technique is based on the heuristic/belief that images are rarely personalized and therefore the presence of a cookie in this case is simply the result of a sloppy design of a Web site.

While HTTP 1.1 explicitly allows caching cookied documents unless prohibited by the content server, and HTTP 1.0 does not contain cookie header at all, in practice, caching these documents would currently result in unpredictable behavior.

4.2 Bandwidth Consumption

An often-cited benefit of proxy caching is reduced network traffic on the Internet. Surprisingly, by carefully modeling the system behavior when requests are aborted, we found that the proxy can actually *increase* the traffic from content providers to the ISP. Without a proxy, when a client aborts a request, the transfer of the document is interrupted and no longer consumes network bandwidth. On the other hand, due to the bandwidth mismatch between the connections from client to proxy and from proxy to the Internet, the proxy may have downloaded much of the document by the time it learns of the abort. Thus the bandwidth consumption of aborted requests is higher when the proxy is present.

The following table indicates the frequency of interrupted transfers for the ISP trace. While some browsers, in response to a user hitting the Stop button, will abort a connection using the RST flags within TCP, others will just close the connection. Therefore, aborts are hard to infer purely based upon the state of the connection. Instead we infer which documents are aborted by relating HTTP response Content-length header fields to the amount of data that was actually received by the clients (excluding header data). This is possible for responses with valid Content-length (i.e. a non-zero value). In the table below the first line corresponds to completed HTTP transfers; the second line in the table indicates incomplete HTTP transfers (about 11% of requests); the third line is an exceptional case—the server sent more data than it claimed as the document's Content-length. Presumably these responses are due to buggy server software.

Case	Count	Percentage	Content-Length Bytes	Response Bytes
CL = RDL	5,487,519	88.79	33731MB	33731MB
CL > RDL	676,243	10.94	44483MB	8210MB
CL < RDL	16,706	0.27	131MB	151MB

The exact effect of aborts on the overall bandwidth demands depends on the handling of aborted requests by the proxy. If the proxy continues the download, subsequent requests to the document will hit in the cache, improving performance. On the other hand, this may further increase the traffic from the content providers to the proxy.

In our simulations, the total number of bytes received by the clients from the content providers without the proxy was 49.8 GB. At one extreme, if the proxy always continued to download upon client's abort, the total number of bytes from the content providers to the proxy would be 58.7 GB (118% of the original number of bytes transferred from the content providers). This means that, with the proxy, one could end up consuming 18% more bandwidth to the Internet.

On the other extreme, the proxy could abort the download immediately after receiving a client's abort. In this case, the the amount of wasted data transmitted depends on the proxy-to-server bandwidth.

The following table compares bandwidth consumption of the network with and without the proxy, for various values of the proxy-to-server bandwidth.

Proxy-server	Data transfered	Savings
45Mbps	53.7GB	-8%
1.5Mbps	50.7GB	-2%
0.5Mbps	41.5GB	15%

As seen from this table, for the network queue bandwidth of 45 Mbps, aborting downloads would reduce wasted data transmissions by only 5.0 GB, still an increase of 8% over the without-proxy case. A 1.5 Mbps Internet connection would save 8.0 GB of wasted data (corresponding to a 2% increase in overall traffic). However, if the bandwidth of the network queue were just 0.5 Mbps[2], savings from caching data would

[2]Note that the lower-bandwidth network queues are appropriate models given that a lot of web sites are accessed over a network path that includes at least one T1 or lower speed connection.

finally offset any additional transfers after abort. The savings would be 15% overall (16.9 GB less unneeded transfers relative to the case where downloads continue after aborts). These bandwidth savings (if any) from the proxy are nowhere near those suggested by the byte hit ratio.

Using the AT&T Labs trace from clients connected to the Internet via a T1 link, we confirmed that the bandwidth mismatch between the modems and the Internet contributes significantly to the wasted bandwidth consumption from aborts. When a well-connected client aborts a transfer, it will have received a significant fraction of the data received by the proxy. Handling of aborts is important even in the case of well-connected clients, however, since if the proxy continues to download the document even after an abort incurs it will see minimal savings. If the proxy aborts immediately it will see 14% savings in bandwidth.

We should note that NetCache attempts to address the aborts problem by implementing flow control over the proxy-to-server connection [3]. The download from the content server to the NetCache is never more than a fixed amount (e.g. 64 KB) ahead of the download from the NetCache proxy to the client. We do not know if any other vendors implement any flow control or address the aborts problem in any other way. It is clear, however, that aborts represent a real issue that must be considered when evaluating the benefits of a proxy cache.

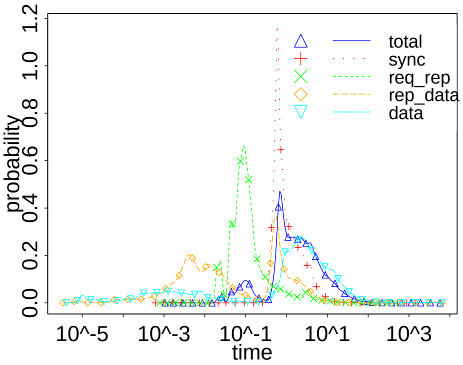
4.3 Latency Reduction: Caching Connections vs. Caching Data

Multiple components within a typical HTTP request contribute to the latency of that request. Depending on the bandwidth environment, adding a proxy cache can influence the latency of any of these components and therefore change the latency experienced by the user in several ways.

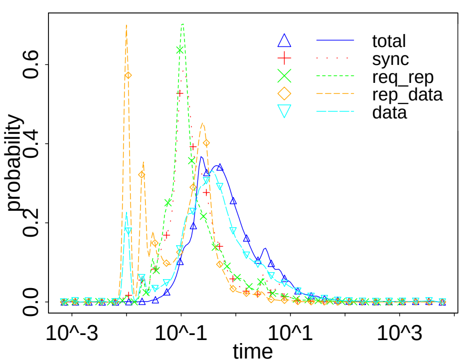
Latency Components

Most document downloads proceed as follows: (1) the client establishes a TCP connection to the server; (2) the client sends the HTTP request for document; (3) the server sends the HTTP response for document; (4) the server sends the data for document; (5) the server/client closes the TCP connection. With persistent connections [12] it is possible to skip steps (1) and (5). In the modem trace (research trace) 18% (12%) of the HTTP requests are sent over persistent connections.

While one would hope that the bulk of the time is spent transferring document data, that is not necessarily the case. Figure 5 illustrates the breakdown of latency components for our original traces without a proxy in



(a) Modem environment



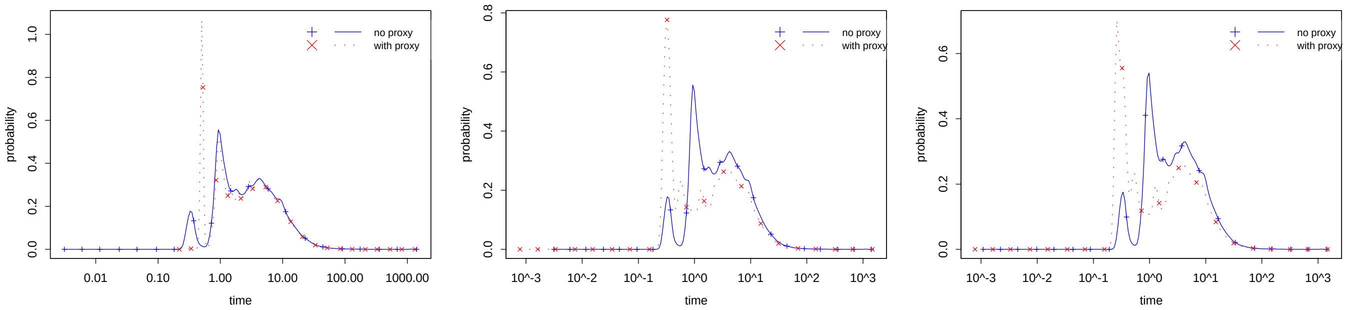
(b) Research Lab environment

Figure 5: *Probability density of total latency and latency components; in both subfigures the curve labeled* total *shows the probability density of the end-to-end request latency; the curve labeled* sync *shows the latency to establish the TCP connection from the client to the content-provider; the curve labeled* req_rep *shows the time between sending the HTTP request to receiving the HTTP response; the curve labeled* rep_data *shows the time between receiving the HTTP response and the first part of the document; finally the curve labeled* data *shows the transfer time for requests in which a document was returned.*

place.[3] Note that connection creation contributes a significant amount of time to the end-to-end latency. The heavy-tailed nature of the document transfer (the mean time is much larger than the median) [2], implies that there are numerous documents (47% for the modem trace and 26% for the research trace) for which the connection setup time is at least 50% of the total download time.

Due to the high cost of TCP connection set-up, potentially significant benefits could be obtained by maintaining persistent connections between clients and servers. However, content-providers can maintain only a limited number of such connections. This suggests that a proxy may also serve in the non-traditional role of a *connection cache*, it would maintain persistent connections with content providers and re-use them for obtaining documents for multiple clients. The connection cache requires only one (or a small number of) persistent connection(s) to a given content provider, regardless of the number of clients connected to the proxy. Similarly, each client needs to maintain only a small number of persistent connections to the proxy regardless of the number of servers it visits. Moreover, if for some reason a connection between the proxy and client or between the proxy and content provider is closed, significant performance benefits could still

[3] The graphs plot the density of the logarithm of the latencies; coupled with a logarithmic scale on the x-axis, plotting the density of the logarithm of the data facilitates direct comparisons between different parts of the graphs based on the area under the curve.



(a) Data cache only (b) Connection cache only (c) Data and connection cache

Figure 6: *Total latency comparison for AT&T WorldNet modem trace*

be obtained due to the connection with the other side.

Modem Trace: Latency Analysis

For the modem trace Figure 6 shows the effects of both serving data from the cache — referred to as data caching — as well as using persistent connections — referred to as connection caching — on the total latency. One would expect that those requests that are served from the cache will be serviced faster. However, we see from Figure 6(a) that caching reduces the latencies experienced by the user by only a minimal margin The two curves fall more or less on top of each other except for the peaks for small latency values in both the with-proxy simulation as well as the no-proxy simulation. The peak in the no-proxy simulation reflects the use of persistent connection in the trace itself. The peak in the with-proxy simulation, at about 500 ms, reflects the minimum amount of time that it takes to establish a connection and exchange the HTTP request/response information and some data given a round-trip time of 250 ms. In total the improvement in the mean and median latencies is 3% and 4% respectively. This result is substantially more pessimistic than the upper bound of 26% latency reduction reduction reported by Kroeger et al. [10]. The limited improvement is due to the heterogeneous bandwidth environment which implies high latency of the connection set-up (which had a mean of 1.3s in the trace) and modem bandwidth limitation on the clients (this study assumed 28.8 Kbps modems). We may be treating the pure data cache unfairly by not even allowing the use of persistent connections for those requests that used persistent connection in the original trace.

Figure 6(b) verifies our intuition that if the proxy is used as a connection cache, the total mean latency improvements grow substantially. In this heterogeneous bandwidth environment the probability shifts from the larger latencies for the no-proxy case to substantially shorter latencies for the with-proxy case. The mean and median latencies improves by 21% and 40% respectively. Indeed, now the first peak of both curves reflects HTTP requests over persistent connections while the second peak in the no-proxy curve reflects the time it takes to establish the TCP connections.

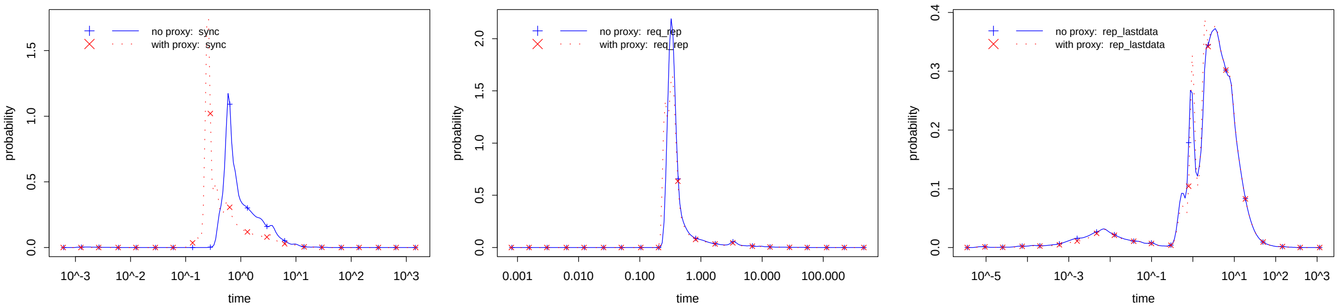
The with-proxy curve of Figure 6(c), which shows the total latency for a Web proxy that is acting both as a connection cache and a data cache, differs only slightly from the with-proxy curve of Figure 6(b). Indeed, the mean and median latencies improves by 24% and 48% respectively. On first thought it may be surprising that this improvement is larger than the individual components would suggest. Data caching can improve the performance of the connection cache by shortening transfers and therefore making a connection available for reuse by a different transfer at an earlier time. This explains why the performance improvements of data caching plus connection caching can be more than additive.

Modem Trace: Latency Analysis Details

To better demonstrate some of the subtleties of data and connection caching, Figure 7 shows the individual components of the total latency of Figure 6(c). The overhead of setting up TCP connections are shown in Figure 7(a). More precisely, it plots the latency between when a client issues a request for a Web document to the time when the Web proxy has received the HTTP request in case of a cache hit or when the Web proxy has sent the HTTP request to the Web server in case of a cache miss. Therefore this graph accurately reflects the savings due to using persistent connections between Web client to Web proxy as well using persistent connections between Web proxy and Web server. The latter is only needed in the case of a cache miss and thus reflects the savings due to data caching.

Figure 7(b) shows the latency that the Web client experiences between sending the HTTP request message and receiving the HTTP response message. The difference between the two curves is due to data caching. In case of a cache hit the Web proxy can send the HTTP response without having to contact the Web server. Unfortunately in the heterogeneous bandwidth environment the latency between the client and the proxy is in the same order of magnitude as the latency between the client and the server, and is not a large proportion of the overall latency. Therefore the benefit of data caching is limited.

The document transfer time (see Figure 7(c)) is the latency between when the Web client receives the HTTP response and when it receives the last data packet. Again the minimal difference between the two



(a) TCP SYN to HTTP request (b) HTTP request to HTTP response (c) HTTP response to DATA end

Figure 7: *Comparison of the latency components for AT&T WorldNet modem trace*

curves is entirely due to data caching. While it may seem surprising that data caching does not significantly improve performance in a modem environment, consider the following cases. If a transfer of a document is limited by the modem bandwidth, a Web proxy can hardly improve the performance except perhaps for the initial startup latency. A comparison of the document transfer latencies and sizes reveals that the modem speed is often a bottleneck. If a transfer of a document is not limited by the modem speed, (typically such documents are small) then a proxy may help. For small documents the latency is determined by the round-trip time. Because the round-trip time is of the same order of magnitude for both cases the difference is small. For large documents it is more unlikely that the document will be cached and the transfer between the proxy and server may be the bottleneck. In this case the fact that the proxy breaks the TCP connection and can buffer some data may have a positive effect. Still, in summary, it is not clear that the benefit of proxy caching on the transfer latency is fairly small for modem clients.

Modem Trace: Caching Connections vs. Caching Data

With the understanding of which latencies of Figure 7 are influenced by data/connection caching, it is clear, that the benefit from connection caching will dominate the total latency improvements for the heterogeneous bandwidth environment. Various connection cache scenarios to consider are: all connections are persistent, only connections from the Web proxy to the Web servers are persistent, only connections from the Web client to the Web proxy are persistent, or none of the connections are persistent. Figure 8 summarizes the performance improvements of various different ways of combining persistent connections with data caching, including cheating on documents with cookies by declaring them cacheable.

Given that a proxy introduces overheads and that the no-proxy simulation uses persistent connections for those requests that were sent across persistent connections in the trace it is not too surprising that using no persistent connections and no data caching will imply an increased latency. Persistent connections between the clients and the Web proxy are not as effective as between the Web proxy and the Web server since they only save the round-trip time between the Web proxy and the Web server. The performance improvements of introducing persistent TCP connections only between the Web server and the Web Proxy are larger than introducing a data cache for documents without cookies and without persistent connections. Adding persistent client connections to caching improves the latency results by more than a factor of 2. Adding all persistent connections improves the latency results over data-caching only by more than a factor of 7.5.

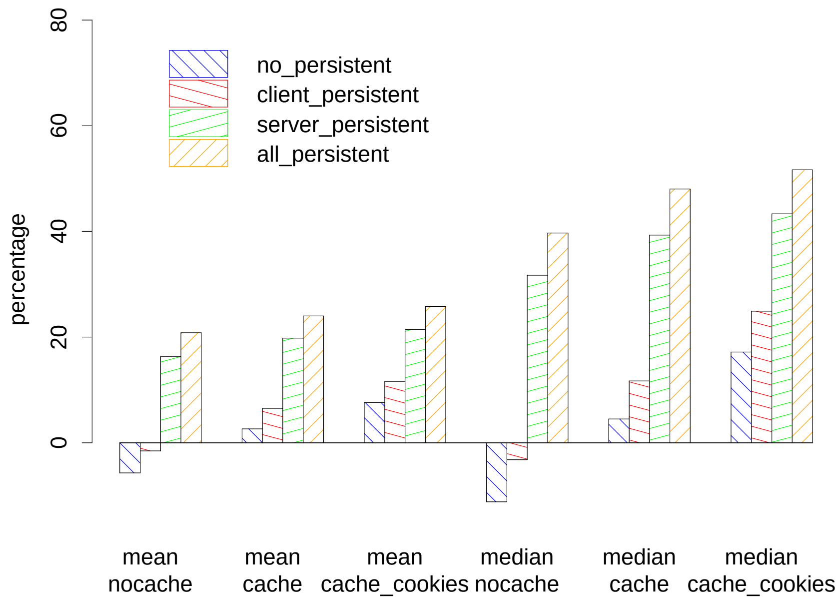
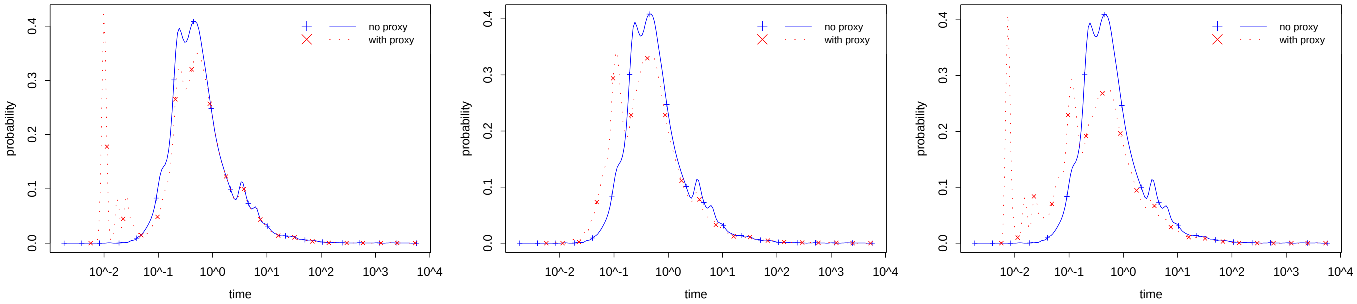


Figure 8: *Latency improvement results for the AT&T WorldNet modem trace relative to the no-proxy case.*

Caching documents improves the median latencies significantly more than the mean latencies since small documents are much more likely to be served from the cache. This means that each individual transfer that uses a persistent connection will occupy the connection for a smaller amount of time. This in turns means that it is more likely that another document transfer can reuse the connection. Note, that enabling caching of documents with cookies will improve the mean latency only from 24% to 26% in the scenario with all persistent connections. A more significant improvement from 3% to 8% can be achieved in the no persistent connection scenario.



(a) Data cache only

(b) Connection cache only

(c) Data and connection cache

Figure 9: *Total latency comparison for AT&T Labs–Research trace*

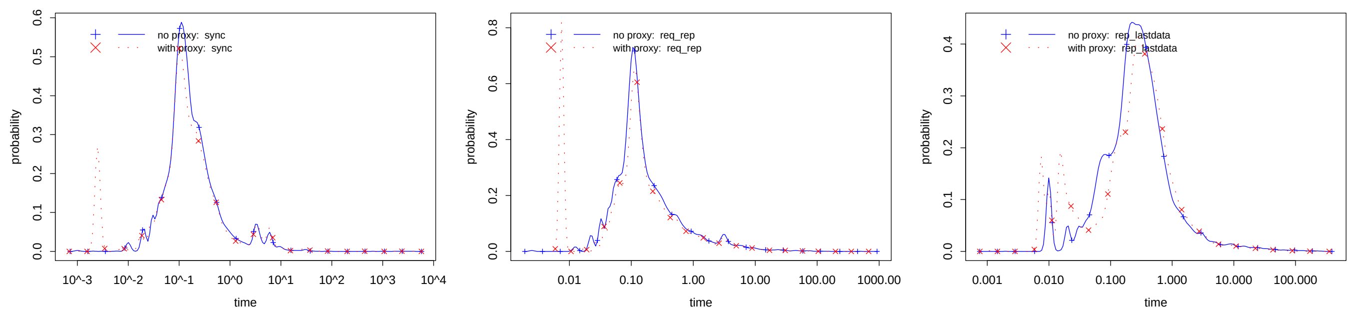
Research Trace: Latency Analysis

Our simulation results for the AT&T WorldNet trace show that connection caching is highly beneficial in a heterogeneous bandwidth environment. To understand the difference between this environment and an environment with a set of well connected clients, Figure 9 shows the total latency comparison for the AT&T Labs–Research trace for the same scenarios as Figure 6.

In this environment we see a mean latency improvement of 7% for the pure data cache proxy (see Figure 9(a)). This is a factor of 2 improvement over the modem environment. On first glance pure connection caching (see Figure 9(b)) yields only a small 2% mean latency improvement compared to the no-proxy results. (The median improves by 20%.) Given that the no-proxy simulation is allowed to use persistent connections for those document transfers that used persistent connections in the original trace this comparison may not be appropriate. Comparing the results to the no caching/no persistent connection case we observe a mean latency improvement of 39% for the research trace, but only an improvement of 8% for the modem trace. When adding connection caching to data caching (see Figure 9(c)) the mean and median latencies improve by 47.5% and 40.4% respectively. This is in the same order of magnitude as the improvements from no persistent connections to all persistent connections.

Research Trace: Latency Analysis Details

Comparing the individual components of the total latencies of Figure 9(c) with those of Figure 6(c) gives insights into how and why the results for the research trace differ from the modem trace. The fact that the area under the first peak in Figure 10(a) is much smaller than for the modem trace (Figure 7(a)) implies that



(a) TCP SYN to HTTP request (b) HTTP request to HTTP response (c) HTTP response to DATA end

Figure 10: *Comparison of the latency components for AT&T Labs–Research trace*

the performance benefit from savings in the connection setup time by itself will much less for the research trace than for the modem trace. The peak is smaller because the round-trip time from the clients to the proxy in the research environment is already small. About half of the peak is due to connection caching and half of it is due to serving data from the cache.

Because of the small round-trip time, the HTTP request to HTTP response latency for cached documents is very small for the research environment (see Figure 10(b)). Another difference in favor of the research trace is that the connection speed to the clients is not a bottleneck. This is reflected in Figure 10(c) where for small documents the with-proxy curve is clearly on top of the no-proxy curve. The cache hit overhead and the interactions between the two TCP connections influence the densities for longer transfers.

All of these effects together contribute to the total user-perceived latency shown in Figure 9.

Research Trace: Caching Connections vs. Caching Data

Figure 11 reflects results for the same scenarios as Figure 8 of combining various forms of connection caching with data caching for the AT&T Labs–Research trace. The results for no cache and no persistent connections show a substantial slowdown because these simulations do have to establish new connections for document transfers that in the original trace and in the no-proxy simulation use persistent connections. It such shows that these requests have a small overall latency.[4] Still introducing a non-caching proxy may slow down Web access even if the Web proxy itself is not a bottleneck. For example a transfer that is delayed by a cache miss may prevent a different transfer from reusing the persistent connection.

[4]We are in the process of simulating the no-proxy without persistent connections and plan to use this as a baseline for the comparison.

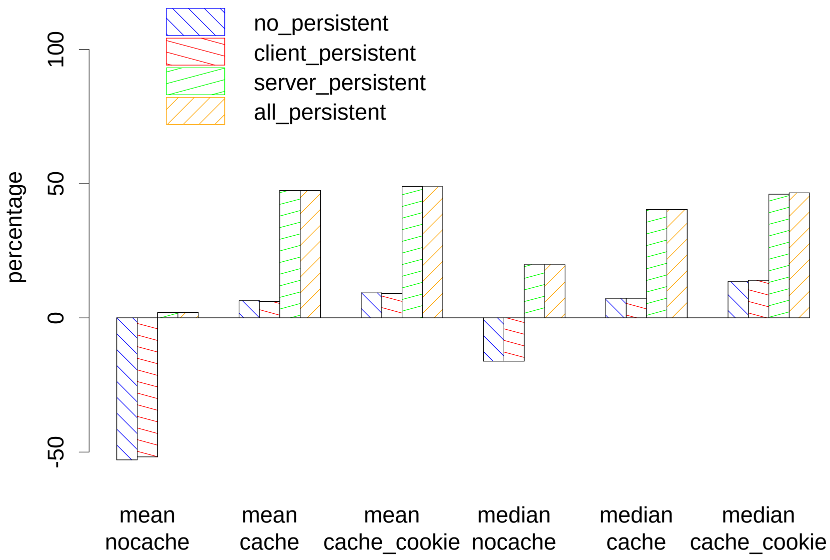


Figure 11: *Latency improvement results for the AT&T Labs–Research trace relative to the no-proxy case.*

Because document caching reduces the document transfer latencies substantially in a low latency environment we see that connection caching and data caching are highly complimentary. Improving the performance by more than 40%. Given that the client to proxy latencies are so small using persistent connections between the Web proxy and the Web server will be sufficient to gain most of the benefit while the benefit of using persistent connection between the Web clients and the Web proxy is almost nonexistent.

Data caching results in substantially better latency improvements (a factor of 2) in the high bandwidth environment, even with a lower hit rate, than in the heterogeneous bandwidth environment. Overall combining data caching with connection caching has a mean and median improvement of 47.5% and 40.5% respectively.

Summary: Caching Connections vs. Caching Data

This analysis of the research and modem traces shows that in a heterogeneous bandwidth environment where the modem is often the latency and bandwidth bottleneck even pure connection caching yields much better latency improvements than pure data caching. In both environments connection caching and data caching are complementary and produce median performance improvements of over 40%.

Connections cached	max # TCP conn to client	max # TCP conn to server	% new conn to client	% new conn to server
all	646	240	18.4	20.6
proxy/server	203	243	100.0	20.8
proxy/client	646	75	18.4	77.8
none	204	73	100.0	77.87

Table 1: Number of connections (modem environment).

Connections cached	max # TCP conn to client	max # TCP conn to server	% new conn to client	% new conn to server
all	216	198	10.1	25.2
proxy/server	179	198	100.0	25.2
proxy/client	216	183	10.1	84.0
none	179	183	100.0	84.0

Table 2: Number of connection (research environment).

4.4 Connection Cache Size

Connection caching can produce substantial performance benefits, thus we need to identify the caching overheads, such as the cache size requirements. *PROXIM* maintains a list of currently active connections. If a connection between a client and a Web proxy is idle for more than 3 minutes it is purged from the cache. A connection to servers are cached for 30 seconds after its last use. The Table 1 and Table 2 summarizes the maximum number of simultaneous connections maintained by a data caching proxy for different connection caching scenarios. The first column of the tables specifies what connections are cached by the proxy. The next two columns contain the maximum connections numbers, which in combination indicate the capacity requirements for the proxy. The last two columns represent the percent of client requests that resulted in opening a new connection, respectively, between a client and the proxy, and the proxy and a server. Note that in the case of proxy-to-server connections, the proxy may avoid opening a new connection by serving the data from the cache or by reusing another connection.

In the modem environment caching connections may increase the maximum number of connections by more than a factor of 3. If this should exceed the facilities of the Web proxy just turning off persistent client connection will reduce the overhead to only a factor of 1.6 while preserving most of the latency benefits. To judge the impact of persistent connection caching on the capacity requirements of the Web server we compare the total number of simultaneous connections without caching and without persistent connections of 112 to the number of connections from the Web proxy to the Web server of 240. An increase of slightly more than a factor of 2.

The costs of using connection caching in the research environment is very small with a factor of less than 1.3. The requirements for the Web servers is actually reduced by a factor of 2 in this case.

The connection cache hit rates are very promising given that the percentage of new connection establishments is reduced by at least a factor of 4 for both environments.

The cost of connection caching can be reduced further by identifying better policies for managing the connection cache. This includes answering questions as when to tear down a connection, which connection to tear down when the proxy runs out of connections, how many simultaneous connections should be maintained to a given content provider or client, when to wait for a persistent connection instead of opening a new connection, and so on.

5 Related Work

From the standpoint of analyzing modem users, the work that is most similar to ours is that of Gribble and Brewer [8]. They collected a trace of HTTP accesses over a modem pool at the University of California, Berkeley, over a period of 45 days and including over 8,000 clients. They used this trace to evaluate several metrics, including the effectiveness of proxy caching and the load placed on servers. We agree with some of their conclusions, particularly the need for efficient support of a large number of TCP connections. We disagree in our enthusiasm with respect to the amount of locality in the clients' reference stream; we do not agree that a "large amount" of locality is present. Clearly there is enough locality to make caching modestly useful, but with cache misses outnumbering (or perhaps barely exceeding) cache hits, the miss case is the common case and needs to be handled well. We believe the difference in our conclusions is due to our assumption that the existence of *cookies* make a resource uncacheable, while they do not mention cookies at all. Since about 30% of the requests in our trace contained cookies, the fraction of inherently uncacheable resources is higher in our trace than in theirs.

Other research on reducing client latency complements our work, particularly to the extent that modem bandwidth is the apparent bottleneck. Kroeger, et al., analyzed the effect of proxy-caching on latency rather than just bandwidth [10]. Using the widely-used Digital trace, they found that proxy caching could reduce latency by only up to 26%, even though the cache hit ratio was roughly 50%. However, their particular study focussed on corporate LANs and WANs, so they did not consider the particular highly disparate environment we have evaluated here. Other approaches to latency reduction include prefetching [17, 10], data compression [14, 15], and delta-encoding [9, 1, 14].

Duska, et al. [5], used a variety of proxy cache logs to evaluate hit rates, locality, sharing, and cache coherence. They found that proxy cache hit rates varied from 24% to 45%, with higher request rates resulting in higher hit rates. They focused on resource hit rates and byte hit rates, but not latency effects.

Persistent connections were first proposed by Padmabhan and Mogul [16] and simulated by Mogul [13]. They are now part of the HTTP/1.1 proposed standard [7]. Mogul's study concerned persistent connections between clients and servers and did not consider connection caching by a proxy. Nielsen, et al. [15] evaluated persistent connections with pipelined requests and found that pipelining dramatically reduced the number of packets transmitted in a sample exchange. Using a variety of server logs, Manley and Seltzer [11] confirmed that persistent connections are generally useful, but questioned whether multiple persistent connections per client are worthwhile given the server resources necessary to support them. Touch et al, similarly argued against TCP connection reuse for slow HTTP connections, because the avoidance of TCP slow-start is unimportant for low-bandwidth links that can have few packets in transit regardless [19]. We argue that our results strongly support persistent connections as a means of reducing end-to-end latency, and the benefits accrue more from the avoidance of TCP connection establishment than from TCP slow-start avoidance.

6 Summary

In this paper, we presented results of a detailed simulation study of Web proxy caching in heterogeneous bandwidth environments. We made three main observations. One, latency reductions from caching persistent connections can be much greater than those from caching data. The two types of caching are complementary and should be used together. Two, the bandwidth saved by caching can be more than offset by the bandwidth wasted by aborted connections. It is important to manage aborted connections well. Three, hit ratios can be severely reduced by the growing percentage of documents containing cookies. More study is needed into effective ways to cache documents with cookies. An important overall lesson was that low-level details have a significant impact on all aspects of system performance: hit ratios, bandwidth utilization, and user-perceived latency. These details should be considered in future studies.

References

[1] Gaurav Banga, Fred Douglis, and Michael Rabinovich. Optimistic deltas for WWW latency reduction. In *Proceedings of 1997 USENIX Technical Conference*, pages 289–303, Anaheim, CA, January 1997.

[2] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of 1996 ACM SIGMETRICS Conference*, 1996.

[3] Peter Danzig, 1998. Personal communication.

[4] Fred Douglis, Antonio Haro, and Michael Rabinovich. HPP: HTML macro-preprocessing to support dynamic document caching. In *Proceedings of the Symposium on Internet Technologies and Systems*, pages 83–94. USENIX, December 1997.

[5] Bradley M. Duska, David Marwood, and Michael J. Feeley. The measured access characteristics of World-Wide-Web client proxy caches. In *Proceedings of the Symposium on Internet Technologies and Systems*, pages 23–35. USENIX, December 1997.

[6] Anja Feldmann, Jennifer Rexford, and Ramón Cáceres. Reducing overhead in flow-switched networks: An empirical study of web traffic. In *Proceedings of 1998 IEEE INFOCOM Conference*, 1998.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, et al. RFC 2068: Hypertext transfer protocol — HTTP/1.1, January 1997.

[8] Steven D. Gribble and Eric A. Brewer. System design issues for internet middleware services: Deductions from a large client trace. In *Proceedings of the Symposium on Internet Technologies and Systems*, pages 207–218. USENIX, December 1997.

[9] Barron C. Housel and David B. Lindquist. WebExpress: A system for optimizing Web browsing in a wireless environment. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking*, pages 108–116, Rye, New York, November 1996. ACM.

[10] Thomas M. Kroeger, Darrell D. E. Long, and Jeffrey C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of the Symposium on Internet Technologies and Systems*, pages 13–22. USENIX, December 1997.

[11] Stephen Manley and Margo Seltzer. Web facts and fantasy. In *Proceedings of the Symposium on Internet Technologies and Systems*, pages 125–133. USENIX, December 1997.

[12] Jeffrey Mogul. The case for persistent-connection HTTP. In *Proceedings of ACM SIGCOMM'95 Conference*, pages 299–313, October 1995.

[13] Jeffrey Mogul. The case for persistent-connection HTTP. *Computer Communication Review*, 25(4):299–313, October 1995.

[14] Jeffrey Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. Potential benefits of delta-encoding and data compression for HTTP. In *Proceedings of ACM SIGCOMM'97 Conference*, pages 181–194, September 1997.

[15] Henrik Frystyk Nielsen, James Gettys, Anselm Baird-Smith, Eric Prud'hommeaux, Hakon Wium Lie, and Chris Lilley. Network performance effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of ACM SIGCOMM'97 Conference*, pages 155–166, September 1997.

[16] Venkata N. Padmanabhan and Jeffrey C. Mogul. Improving http latency. *Computer Networks and ISDN Systems*, 28(1–2):25–35, December 1995.

[17] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve World Wide Web latency. *Computer Communication Review*, 26(3):22–36, 1996.

[18] Squid internet object cache. http://squid.nlanr.net/, 1997.

[19] Joe Touch, John Heidemann, and Katia Obraczka. Analysis of http performance. USC/Information Sciences Institute. Available as http://www.isi.edu/lsam/publications/http-perf/.