Network-Level Impacts on User-Level Web Performance

Carey Williamson   Nayden Markatchev
Department of Computer Science, University of Calgary
Email: {carey,nayden}@cpsc.ucalgary.ca

Keywords: Web Performance, Web Proxy Caching, Web Caching Hierarchies, TCP, ns2

Abstract

This paper uses packet-level ns2 network simulations to study the impacts of TCP and network-level effects on user-perceived Web performance in Web proxy caching hierarchies. We model a simple two-level Web proxy caching hierarchy, and study the effects of link speed, propagation delay, request arrival rate, cache hit ratio, and cache management policy on the transfer latency for Web document downloads. The multi-factor experiments show that link capacity, round-trip time, and TCP behaviours all have a significant influence on user-level response time. The simulation results also highlight the relationships between cache hit ratio and user-perceived Web performance.

INTRODUCTION

The Web is both a blessing and a curse. The blessing is that it has made the Internet available to the masses. Through the information-hiding principles of the layered Internet protocol stack, Web users at the application layer need not know the technical details about the underlying communication protocols. The Web provides seamless exchange of information, in a location-independent, time-independent, and platform-independent manner.

The curse, however, is that the popularity of the Web has placed a lot of stress on the Internet, both in traffic volume and in demand on the TCP/IP protocols. Web users expect low latencies for interactive Web browsing sessions, regardless of the sizes of the objects retrieved, or the locations from which they are retrieved. A wealth of Web caching research [5, 8, 10, 11, 13, 14, 15, 17, 21, 25] has addressed the traffic volume issue, while protocol research has addressed TCP/IP and HTTP performance issues [1, 3, 6, 7, 20, 23, 24, 26].

Earlier work in the literature has highlighted some of the mismatches between TCP and the Web [20, 24, 26]. For example, TCP's three-way handshake and the slow start algorithm [3] each add latency to Web document transfers. In addition, the bursty nature of packet transmissions in TCP flow control can lead to packet losses in a congested network. Recovery from these packet losses can add considerable latency to a TCP transfer, particularly for small Web document downloads [26].

Several mechanisms have been proposed to improve TCP performance for the Web. These include concurrent (parallel) TCP connections in most Web browsers, the management of multiple TCP connections as an ensemble [16], persistent-connection HTTP [23], larger initial window sizes for TCP slow start [2], a "fast start" mechanism to reduce transfer latency [24], rate-based pacing of TCP packet transmissions [1, 20], and a context-aware version of TCP that minimizes the impact of packet losses on Web document transfers [26].

Regardless of the mechanisms used, user-level Web performance is highly sensitive to the packet-level dynamics of the TCP protocol. The purpose of this paper is to demonstrate these effects in a Web proxy caching hierarchy. We strive to show the impacts of link capacity, propagation delay, and TCP on Web response time.

Our work uses packet-level simulations in ns2 [9]. We model a simple two-level Web proxy caching hierarchy, and study the transfer latencies for Web document downloads when the characteristics of the network topology and Web workload are varied. The main observation is that cache hit ratios do not tell the whole story: link capacity, round-trip time, network congestion, and TCP behaviours all influence end-user response time significantly.

BACKGROUND AND RELATED WORK

Much of the existing literature on Web performance falls into one of two categories. In the first category are Web caching simulation studies, which typically use cache hit ratio or byte hit ratio (or some combination of the two) as the primary performance metric [5, 11, 14]. These papers tend to ignore the impacts of network-level effects and TCP protocol behaviours on end-to-end Web performance, focusing solely on application-level Web caching performance. In the second category are TCP performance papers, which do consider packet-level behaviours, but often have simplifying assumptions about the user-level workload (e.g., bulk transfers) carried by the TCP protocol. Few papers, except for those by Feldmann *et al.* [17, 18], consider both application-level and network-level issues.

The primary motivation for our current paper comes from some of our own earlier work, by Busari and Williamson [11]. In particular, Busari *et al.* propose the notion of "heterogeneous" Web proxy caching hierarchies, wherein different levels of the caching hierarchy use different cache replacement policies, or size-based thresholding to separate the Web content cached at each level of the hierarchy. In [11], the authors speculate:

> "Determining which policy is 'best' . . . requires in-depth consideration of network capacity, network latencies, server load, HTTP, and TCP-level effects."

Our present paper starts from this point and explores the relationships between Web caching, network-level effects, TCP, and user-perceived Web performance.

EXPERIMENTAL METHODOLOGY

Simulation Model and Assumptions

The network model for the simulation study is shown in Figure 1. The model assumes a set of Web Clients accessing static Web content from an origin Web Server via a two-level Web proxy caching hierarchy.

Conceptually, requests generated by the Web Clients are sent to the Child Proxy, over network link L_1 which has capacity C_1 and propagation delay d_1. If the request is a "hit" at the Child Proxy (with Hit Ratio HR_1), the content is returned directly to the client over link L_1. If the request is a "miss" at the Child Proxy, then the request is forwarded to the Parent Proxy over link L_2, with capacity C_2 and propagation delay d_2. If the request is a "hit" at the Parent Proxy (with Hit Ratio HR_2), the content is returned to the client over links L_2 and L_1. Otherwise, the request is forwarded to the origin Web Server over link L_3, which has capacity C_3 and propagation delay d_3. The origin Web server always has the target content for these requests, and thus returns the content to the client over links L_3, L_2, and L_1.

There are two separate stages in our simulation process. First, an application-level Web caching model is used for the Web proxy caching hierarchy, for a given input Web workload. This could be a trace-driven simulation with a Web caching simulator like WebTraff [22], to determine which requests are hits and misses at each level of the Web caching hierarchy, according to the cache sizes and cache replacement policies used. We use a simpler approach, with a script to choose probabilistically (at random) the hits and misses, according to the desired cache hit ratios. Second, an ns2 [9] network simulation is used to model the TCP transfers on the simulated network topology in Figure 1. All data transfers flow from left to right in the diagram, using a 512-byte TCP segment size for data packets. The source of each transfer (either the Web Server, the Parent Proxy, or the Child Proxy) is determined from the first simulation stage. The destination for each transfer is always the Web Clients. The TCP Reno protocol model in ns2 is used to model these transfers, using the TCP flow control and congestion control mechanisms. Note that only the data transfer phase of TCP is modeled; TCP connection setup and teardown are not modeled. Furthermore, the ns2 simulation stage treats the Web proxy caches as routers: these nodes do store-and-forward at the packet level to model cut-through forwarding at the document level for the Web proxy. Each router has a buffer size of 100 packets.

The output from the second simulation stage reports the transfer latency in seconds for each Web document download initiated by the Web Clients. The results from this stage are the main focus of this paper. Simulation results for Web proxy cache performance (e.g., document hit ratio and byte hit ratio for different cache sizes and replacement policies) are presented in earlier papers [10, 11].

Table 1. Experimental Factors and Levels

Factor	Levels
Link Capacity C (Mbps)	**10**, 100, 1000
Propagation Delay d (msec)	**1**, 5, 10, 30, 60
Arrival Rate λ (requests/sec)	**10**, 20, 40, 80
Child Proxy Hit Ratio HR_1	20%, 30%, **40%**
Parent Proxy Hit Ratio HR_2	7%, 10%, **15%**

Experimental Design

Table 1 summarizes the experimental factors and levels used in the experiments. The characteristics of the network topology are changed using two factors: link transmission capacity C, and link propagation delay d. The latter value affects the round-trip time (RTT) for TCP connections. Two workload factors are also varied: the request arrival rate, and the hit ratios at each of the proxies in the Web caching hierarchy. These factors affect the utilization of the network links. Values in **bold** font in Table 1 represent settings used in the baseline scenario.

A full-factorial experimental design [19] is used, to explore every possible combination of factors and levels. A total of 192 ns2 simulations are conducted. Only a small subset of selected results appear in the paper.

The primary performance metric used is the transfer latency in seconds for each Web document download. Most analyses focus on the transfer latency as a function of transfer size (either in bytes or in packets).

Web Workload

The workload for the simulation experiments consists of 5000 HTTP transfers. The HTTP transfers were selected from a longer Web workload trace, which was synthetically generated using the WebTraff tool [12, 22]. This modest trace length was chosen because of the relatively high overhead of packet-level network simulation, compared to Web proxy caching simulation. The resulting sequence of 5000 transfers is deemed to represent steady-state cache performance for a simple two-level Web proxy caching hierarchy, using Least-Recently-Used (LRU) replacement policies. Table 2 summarizes the workload used.

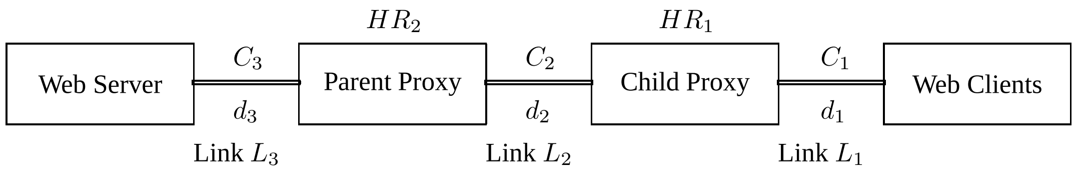


Figure 1. Network Model for Simulation Study

In the packet-level simulation experiments, each HTTP transfer is modeled as a TCP Reno data transfer in ns2. The requests arrive according to a Poisson arrival process with a specified arrival rate λ. Four different arrival rates are considered: a "Light" load of 10 requests per second, a "Moderate" load of 20 requests per second, a "Medium" load of 40 requests per second, and a "Heavy" load of 80 requests per second. These workloads represent offered loads ranging from 0.8 Mbps to 6.1 Mbps.

Warmup and Run-Length Issues

We assess simulation "steady-state" based on the number of simultaneously active TCP connections in the simulated network. Because the TCP transfers vary in duration (depending on number of packets, round trip time, network congestion, packet losses, and retransmissions), the number of active TCP connections in the simulated network varies with time. These concurrent TCP connections compete with each other for available network resources, based on the TCP flow control and congestion control algorithms.

Preliminary experiments showed that the time required to reach steady-state is brief (e.g., a few seconds). The light load scenario generates up to 8 TCP connections in the network at a time. For heavy load, up to 43 connections are active simultaneously. The simulation "end effects" also have modest impact. For simplicity, all performance results in the rest of the paper are calculated for all 5000 transfers in the simulation.

SIMULATION RESULTS

This section presents the simulation results. We start with an overview of the results for the baseline scenario, and then methodically consider the impacts of link transmission capacities, propagation delays, request arrival rate, cache hit ratio, and cache management policy.

Baseline Scenario

The first experiment considers the baseline scenario. The network model for this scenario is shown in Figure 1, and the parameter settings are highlighted in bold font in Table 1. The baseline scenario assumes that the link transmission capacities are $C_1 = C_2 = C_3 = 10$ Mbps. The propagation delays are $d_1 = 1$ msec to represent a Child Proxy in a Local Area Network (LAN) near the Web Clients, $d_2 = 5$ msec to represent a Parent Proxy in a regional or Metropolitan Area Network (MAN) by the Web Clients, and $d_3 = 30$ msec to represent an arbitrary Internet Web Server on the Wide Area Network (WAN). The request arrival rate is $\lambda = 10$ requests/sec. The cache hit ratio at the Child Proxy is $HR_1 = 40\%$. That is, approximately 40% of the 5000 HTTP transfers are satisfied by the Child Proxy. The cache hit ratio at the Parent Proxy is $HR_2 = 15\%$. Note that this means that 15% of the 5000 client requests[1] are satisfied at the Parent Proxy. A lower hit ratio is assumed at the Parent Proxy because of the filtering effect of the first-level cache [11, 15, 25]. The remaining 45% of the HTTP requests are satisfied by the origin Web Server.

Figure 2 provides an overview of the simulation results for this baseline scenario, using a scatter plot of transfer

Table 2. Simulated Web Proxy Workload

Item	Value
Total Transfers	5,000
Total Transferred Bytes	48,121,956
Smallest Transfer Size (bytes)	68
Median Transfer Size (bytes)	3,958
Mean Transfer Size (bytes)	9,624
Largest Transfer Size (bytes)	1,494,900
Web Server Transfers	2,252
Web Server Hit Ratio	45.0%
Web Server Byte Hit Ratio	43.6%
Total Bytes Served (Web Server)	20,957,250
Mean Transfer Size (bytes)	9,306
Parent Proxy Transfers	758
Parent Proxy Cache Hit Ratio	15.2%
Parent Proxy Cache Byte Hit Ratio	14.2%
Total Bytes Served (Parent Proxy)	6,821,625
Mean Transfer Size (bytes)	9,000
Child Proxy Transfers	1,990
Child Proxy Cache Hit Ratio	39.8%
Child Proxy Cache Byte Hit Ratio	42.3%
Total Bytes Served (Child Proxy)	20,343,081
Mean Transfer Size (bytes)	10,223

[1]Technically, this is not the proper definition of the cache hit ratio, since only a subset of the requests from the Web Clients are seen at the Parent Proxy. However, this abuse of terminology simplifies the description of our simulation model.

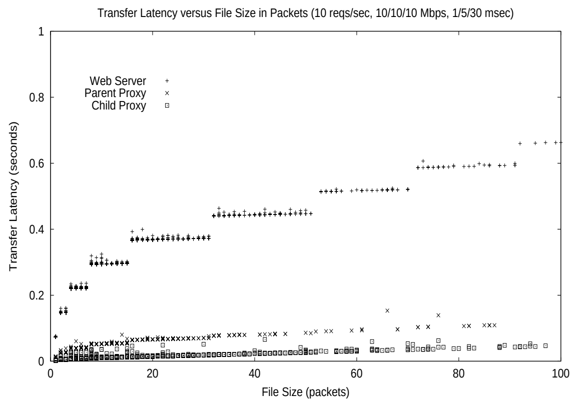


Figure 2. Simulation Results for Baseline Scenario

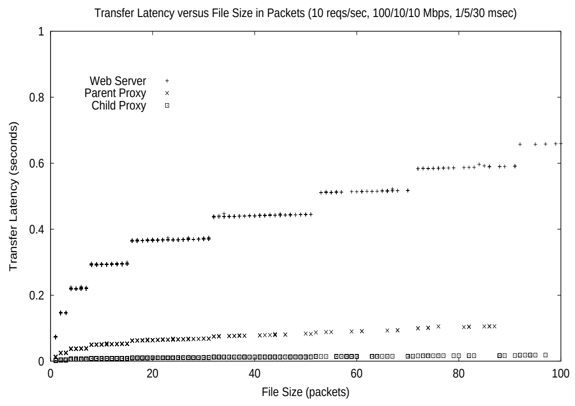


Figure 3. Effect of Link Capacity $C_1 = \mathbf{100}$ Mbps

latency (in seconds) versus transfer size (in packets). Each dot on the graph represents one transfer.

In Figure 2, the transfer latencies exhibit a distinct structure, with three lines. The uppermost line represents WAN transfers from the Web Server: the average slope corresponds to about 1 Mbps. The lowermost line corresponds to transfers from the Child Proxy. These transfers have an average throughput of about 8 Mbps. The middle line corresponds to transfers from the Parent Proxy; these average about 6 Mbps. The throughputs depend on the round-trip time (RTT) for LAN, MAN, and WAN transfers.

The step-like structure in the transfer latency plot represents the TCP slow start algorithm. In TCP slow start, a source is allowed to send 1 TCP segment (i.e., data packet) in the first RTT, 2 in the second RTT, 4 in the next RTT, and so on, as long as per-packet acknowledgements are returned successfully. The graph shows that the "width" of the steps increases exponentially, according to the TCP slow start algorithm (i.e., a congestion window size of 1, 2, 4, 8, and 16 packets). The consistent vertical spacing between the steps corresponds to the RTT (72 msec for the WAN, 12 msec for the Parent Proxy).

Figure 2 also classifies the data points based on the source of each transfer: Web Server ('+'), Parent Proxy ('X'), and Child Proxy (boxes). The bulk of the Child Proxy transfers cluster along the lowermost line in the graph, though there are a few points above this line. In general, the deviations reflect queueing delays, packet losses, timeouts, and retransmissions, which end up affecting the TCP transfer latency. Similar observations apply for the Parent Proxy and Web Server transfers.

The observations stated here for the baseline scenario hold in general for many of the other scenarios studied. For space reasons, the remaining analyses in this paper focus primarily on the scatterplot of TCP transfer latency versus transfer size in packets (i.e., the format in Figure 2), which illustrates performance differences as network and workload parameters are varied.

Effect of Network Link Capacity

The second experiment varies the link transmission capacities C_i in Figure 1. We start with the baseline scenario of 10 Mbps links everywhere, and then upgrade the client access link L_1 to $C_1 = 100$ Mbps. We follow this with an upgrade of the proxy link L_2 to $C_2 = 100$ Mbps, and then a further upgrade of the client access link L_1 to $C_1 = 1$ Gbps. In all cases, the link L_3 to the origin Web server has $C_3 = 10$ Mbps. This range of scenarios represents clients with increasingly faster access to an Internet with a slower WAN backbone link. We believe this set of scenarios is representative of today's Internet.

Recall that the baseline scenario in Figure 2 shows evidence of queueing delays for some transfers, regardless of their point of origin. These observations suggest that the bottleneck is at the client access link L_1, since all transfers traverse this link en route to the Web Clients. Direct calculation shows that the average traffic load offered to link L_1 in the baseline scenario is 0.8 Mbps, representing an average utilization of less than 10% for this 10 Mbps link. However, the peak utilization is typically much higher than the mean, and the burstiness of the TCP packet arrival process can still induce queueing, even at this light load. No packets were lost in this scenario.

Figure 3 shows the results when the client access link L_1 is upgraded to $C_1 = 100$ Mbps. Here, the network bottleneck is alleviated substantially. The transfer latencies from the Child Proxy have almost a perfectly linear relationship with transfer size, indicating that queueing delays, packet losses, and retransmissions are not a problem. The network congestion is also alleviated for transfers from the Parent Proxy and the Web Server, though there are still a few small deviations for Web Server transfers. Separate experiments (not shown here) show that upgrading the proxy link L_2 to $C_2 = 100$ Mbps offers no further improvement in the transfer latencies, since the remaining bottleneck is further upstream at the Web Server's link. Similarly, upgrading the client's link to 1 Gbps offers no improvement.

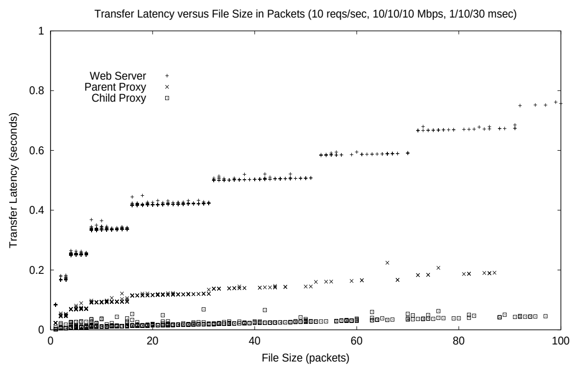


Figure 4. Effect of Propagation Delay $d_2 = \mathbf{10}$ msec

The results in Figure 3 establish two key points. First, the bottleneck in the baseline scenario is at the client link L_1. Even at the lightest load considered (10 requests/sec), there is sufficient demand on this link to induce queueing. Second, there is little point considering L_1 link capacities exceeding $C_1 = 100$ Mbps. This higher capacity is adequate to eliminate the bottleneck.

Effect of Network Propagation Delay

The next experiment considers the impact of link propagation delay (and thus round-trip time, RTT) on the TCP transfer performance. We start with the baseline scenario, where all link capacities are 10 Mbps, with $d_1 = 1$ msec (LAN), $d_2 = 5$ msec (MAN), and $d_3 = 30$ msec (WAN). We then "stretch" the network to model a Parent Proxy that is farther away ($d_2 = 10$ msec), a Web Server that is farther away ($d_3 = 60$ msec), and then a combination of these two conditions. In all cases, we assume that the Child Proxy is close to the Web Clients ($d_1 = 1$ msec). We believe that this set of scenarios represents typical Web proxy use on the Internet.

Figure 4 presents the simulation results for a Parent Proxy farther away ($d_2 = 10$ msec) from the Child Proxy. In this graph, there is a larger vertical separation between the transfer latencies for the Parent Proxy and the Child Proxy. (The transfer latencies for the Web Server case have moved up as well, since these transfers also traverse link L_2.) This additional vertical separation corresponds to the greater RTT for transfers from the Parent Proxy. These results are as expected. In essence, the larger RTT increases the "cache miss penalty" for misses at the Child Proxy. The larger RTT (22 msec) for Parent Proxy transfers makes the step-like structure of transfer latency more apparent, though this structure is still partly masked by the congestion effects at link L_1.

To summarize this experiment, the network link delays determine the vertical spacings in the distinct step-like timing structure of the transfer latencies. This structure is most evident for transfers from the Web Server. The set of propagation delays chosen in the baseline scenario is adequate for illustrating this structure, and is arguably adequate for representing the range of RTT delays commonly experienced on the Internet.

Effect of Request Arrival Rate

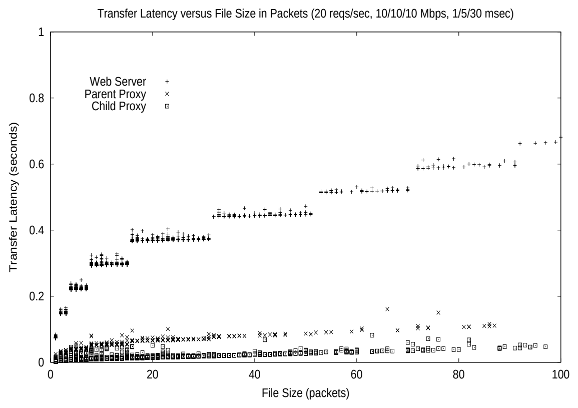
The next experiment varies the request arrival rate for the simulated Web workload. We use the same set of requests, in the same order, but simply reduce the interarrival times between requests to achieve higher target request rates. We consider request rates λ ranging from 10 requests/sec to 80 requests/sec.

Figure 5 presents the simulation results from this experiment. From the baseline scenario in Figure 2, the request arrival rate is progressively doubled to 20 requests/sec in Figure 5(a), 40 requests/sec in Figure 5(b), and 80 requests/sec in Figure 5(c). Given these arrival rates, the average traffic load offered to the 10 Mbps bottleneck link L_1 by these four scenarios is 0.8 Mbps, 1.5 Mbps, 3.0 Mbps, and 6.0 Mbps, respectively. The latter is a heavy load for the bottleneck link L_1.

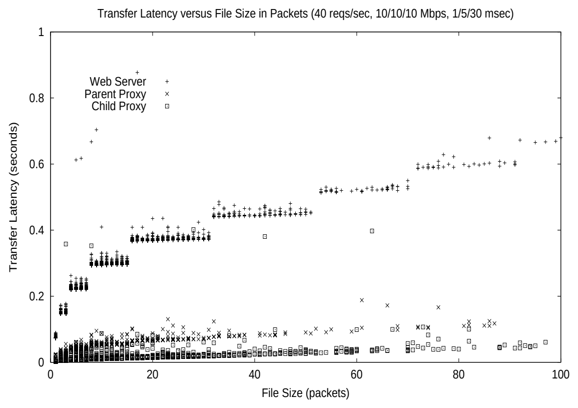
Figure 5(a) shows that doubling the arrival rate to 20 requests/sec increases the congestion that was first observed in the baseline scenario. More points are displaced upward from the step-like structure of transfer latency, with some points displaced much further than before. This behaviour is most evident for Parent Proxy transfers of 5 to 30 packets, and for Web Server transfers of 20 to 30 packets.

The increasing congestion trend is even more evident in Figure 5(b), where the request rate is 40 requests/sec. Here, there is a vertical "blurring" of the transfer latencies for Web Server transfers, particularly for small transfers (less than 20 packets). There is also a "blending" of transfer latencies for Parent Proxy and Child Proxy transfers, for small transfers (e.g., less than 15 packets). In other words, the congestion on the client link L_1 adds enough queueing delay to make Child Proxy hits visually indistinguishable (at least on the graph) from Parent Proxy hits. Transfers from the Web Server are also affected: there is a small cluster of points in the upper left portion of the graph that are indicative of TCP timeouts and retransmissions. (In addition, 4 data points exceed the 1.0 second upper bound of the graph. One of these is 6.005 seconds; it represents a 3-packet transfer that lost its first data packet. The default TCP timeout of 6 seconds in ns2 makes for a painful recovery from this packet loss.)

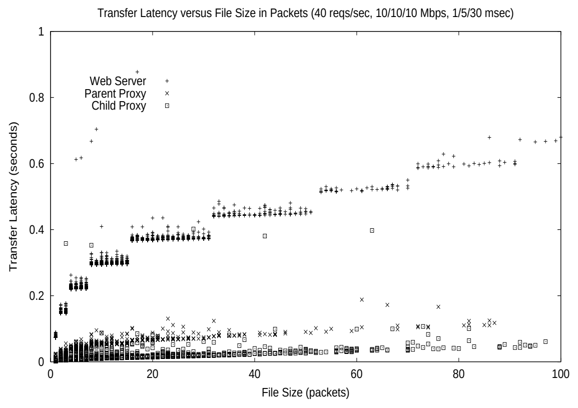
In Figure 5(c), the effects of network congestion are even more rampant. There is a secondary "band" of Child Proxy transfer latencies (the boxes on the graph) near 0.4 seconds, indicating the effects of TCP packet losses, timeouts, and retransmissions. The packet loss problem is particularly acute for small transfers, since the TCP congestion window size is rarely large enough to enable TCP's fast retransmit algorithm. Elsewhere on the graph, there is a small band of X's near 0.5 seconds for the Parent Proxy transfers,



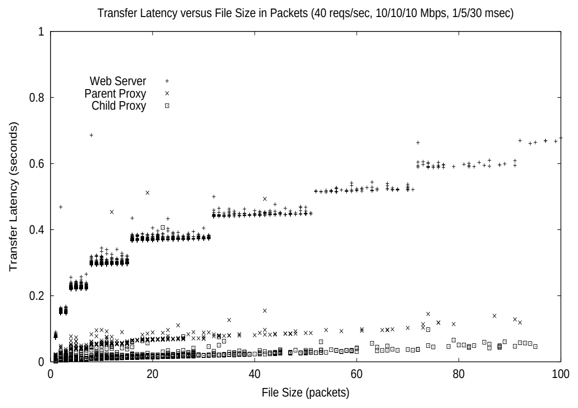
(a) $\lambda = \mathbf{20}$ requests/sec



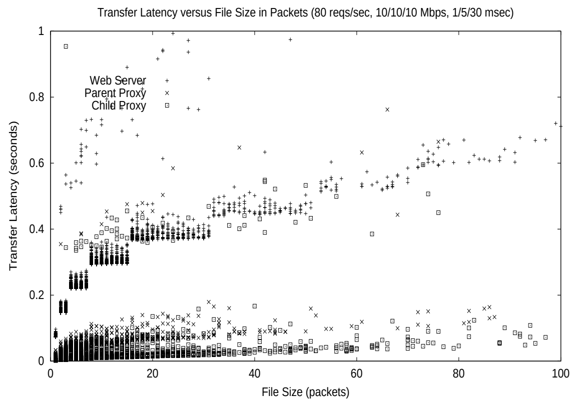
(a) $HR_1/HR_2 =$ 40%/15%, $C_1 = 10$ Mbps



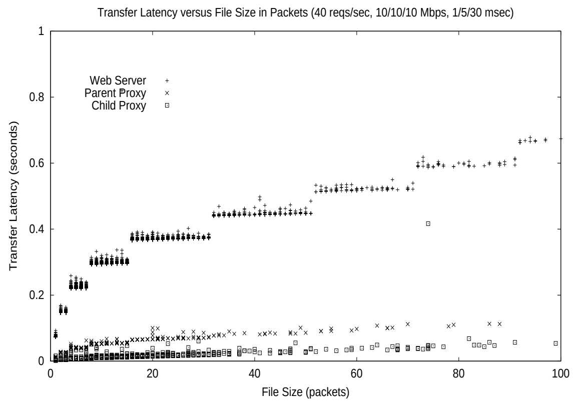
(b) $\lambda = \mathbf{40}$ requests/sec



(b) $HR_1/HR_2 =$ **30**%/**10**%, $C_1 = 10$ Mbps



(c) $\lambda = \mathbf{80}$ requests/sec



(c) $HR_1/HR_2 =$ **20**%/**7**%, $C_1 = 10$ Mbps

Figure 5. Effect of Request Arrival Rate λ

Figure 6. Effect of Cache Hit Ratio HR ($\lambda = \mathbf{40}$ req/sec) and a band of +'s for the Web Server transfers in the upper left portion of the graph. There are 58 transfers that exceed 1.0 seconds in duration, with 27 of these requiring 6 seconds or more.

There is an obvious increase in the mean transfer latency as the request rate is increased, regardless of the source of the TCP transfer. These results are as expected.

The simulation results in this experiment indicate that the 80 requests/sec scenario saturates the bottleneck link. We thus use $\lambda = 40$ requests/sec in the remaining experiments to represent a busy but not overloaded network.

Effect of Cache Hit Ratio

The next experiment studies the impact of the cache hit ratio (HR) on the HTTP transfer latency. In particular, we vary the hit ratio HR_1 at the Child Proxy and the hit ratio HR_2 at the Parent Proxy. In our experiments, these parameters specify the average proportion of requests from the Web Clients that are satisfied at the respective proxy. We vary the two parameters in concert, with HR_2 typically set to be about one-third of HR_1, to reflect the filtering effect of the Child Proxy cache [11, 15, 25].

We consider three cases in this experiment: a "Good" proxy scenario (the baseline scenario), where $HR_1 =$ 40%, $HR_2 =$ 15%, and $HR_{total} = HR_1 + HR_2 = 55\%$; an "Average" proxy scenario, where $HR_1 =$ 30%, $HR_2 =$ 10%, and $HR_{total} = 40\%$; and a "Poor" proxy scenario, where $HR_1 =$ 20%, $HR_2 =$ 7%, and $HR_{total} = 27\%$. We believe that these scenarios span the relevant range for Web proxy cache performance on the Internet [21].

Figure 6 shows the results from this experiment. Figure 6(a) repeats the earlier results for the "Good" cache hit ratio scenario from Figure 5(b), for easy reference. To recap, some network congestion is evident on link L_1.

Figure 6(b) shows results for the "Average" cache hit ratio scenario. Some effects of network congestion are still evident in Figure 6(b), though surprisingly they are not as pronounced as in Figure 6(a). Note that Figure 6(b) has more points in the Web Server transfer case than in Figure 6(a), because of the lower values for HR_1 and HR_2.

Figure 6(c) extends the analysis to the "Poor" Web caching scenario. Other than the increasing number of data points for Web Server transfers, the main observation here is that the effect of network congestion on the transfer latencies is diminishing.

The surprising observation in Figure 6 is the *diminishing* impact of network congestion as the Web caching performance gets worse. Similar observations apply for the 80 requests/sec workload scenario (not shown here, for space reasons), where the diminishing effect is even more pronounced. These results are counter-intuitive, since a lower cache hit ratio typically implies *greater* load on the network links leading to the Internet.

The explanation for this counter-intuitive result requires careful consideration of the bottleneck link L_1, as the cache hit ratio parameters are varied. In particular, when more requests are "pushed" upstream to be handled by the Web Server, the packet arrival process at the bottleneck link L_1 is fundamentally changed. While the same number of TCP data packets must eventually traverse the bottleneck link to reach the Web Clients (i.e., in order to complete the TCP transfer successfully), the arrivals of these packets are now *spread out in time* (i.e., the same number of packets are transmitted, but over a longer connection duration, because of TCP flow control and the larger RTT). Because there are fewer Child Proxy transfers (with short 2 msec RTT) competing for the bottleneck link, there is less queueing and packet loss there. Furthermore, the few remaining Child Proxy transfers in the workload tend to complete *sooner* than in the congested network case, freeing up link bandwidth for use by other connections. Table 3 summarizes the transfer latency results for this experiment.

Table 3. Cache Hit Ratio Simulation Results

Item	Good	Average	Poor
Total HTTP Transfers	5,000	5,000	5,000
Mean Latency (sec)	**0.156**	**0.191**	**0.229**
Web Server Transfers	2,252	2,973	3,642
Mean Latency (sec)	0.306	0.302	0.303
Parent Proxy Transfers	758	514	354
Mean Latency (sec)	0.058	0.058	0.055
Child Proxy Transfers	1,990	1,513	1,004
Mean Latency (sec)	**0.025**	**0.018**	**0.022**

In summary, this experiment has illustrated an interesting performance tradeoff related to Web caching. While the overall mean transfer time *increases* (as expected) when the hit ratio is lower (since more transfers have to contact the Web Server), the mean response time for Child Proxy cache hits *decreases*, since there is less contention for the bottleneck link. That is, while there are *fewer* Child Proxy transfers, they are *faster*, and have lower variance. The variance of the transfer latency for Parent Proxy and Web Server transfers is also reduced, because the congestion on link L_1 is alleviated. However, congestion at link L_3 may become more of a problem.

Effect of Cache Management Policy

The final simulation experiment addresses an open issue identified in earlier work [11] on multi-level caching hierarchies. In particular, a size-based thresholding scheme was proposed for a two-level caching hierarchy, where only "small" files were eligible for caching at the first level of the hierarchy, and only "large" files were eligible for caching at the second level. The primary advantage of this approach is that it partitions the Web document space across the caches, eliminating duplication of content in the Parent Proxy and the Child Proxy. A secondary advantage

Table 4. Workload for Size Threshold Experiments

Item	Value
Total HTTP Transfers	5,000
Total Transferred Bytes	48,121,956
Web Server Transfers	2,187
Web Server Hit Ratio	43.74%
Web Server Byte Hit Ratio	44.37%
Total Bytes Served (Web Server)	21,353,484
Mean Transfer Size (bytes)	9,764
Parent Proxy Transfers	723
Parent Proxy Cache Hit Ratio	14.5%
Parent Proxy Cache Byte Hit Ratio	42.2%
Total Bytes Served (Parent Proxy)	20,291,324
Mean Transfer Size (bytes)	**28,065**
Child Proxy Transfers	2,090
Child Proxy Cache Hit Ratio	41.8%
Child Proxy Cache Byte Hit Ratio	13.5%
Total Bytes Served (Child Proxy)	6,477,140
Mean Transfer Size (bytes)	**3,099**

is providing faster access to content for small Web transfers, since they are closer to the clients. The latter advantage was based solely on an intuitive argument by the authors in [11].

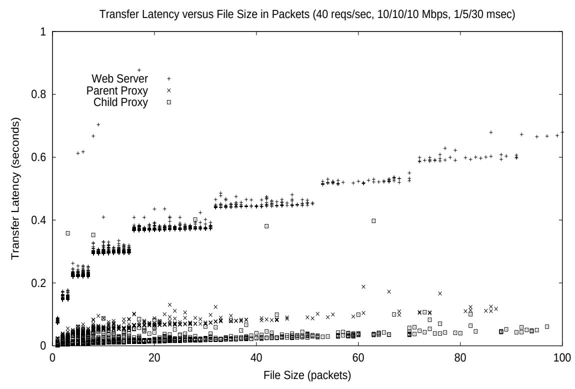
In our final experiment, we quantify the performance advantages (or disadvantages) of size-based thresholding. Table 4 shows the workload used for this experiment.

Figure 7 shows the results from this simulation experiment, using 8 KB as the threshold between “small” and “large” Web transfers. The experiments are conducted for the baseline scenario (before and after an upgrade to link L_1), assuming an arrival rate of $\lambda = 40$ requests/sec.

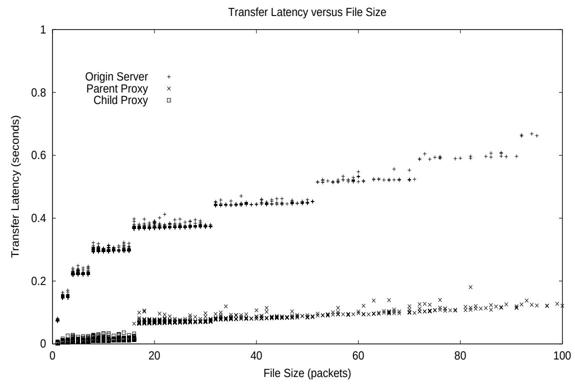
Figure 7(a) repeats the previous results for $\lambda = 40$ requests/sec, for $C_1 = 10$ Mbps. This graph assumes the default caching policy, wherein the Parent Proxy and the Child Proxy are both eligible to cache any Web document.

The graph in Figure 7(b) presents results for the size-based thresholding policy. The Child Proxy is only eligible to cache small documents (at most 8 KB), while the Parent Proxy is only eligible to cache large documents (larger than 8 KB). For ease of comparison with earlier results, we assume $HR_1 = 40\%$ and $HR_2 = 15\%$. (That is, we ignore the obvious fact that the Child Proxy could store *more* documents than in the baseline case, since the cached documents are smaller.)

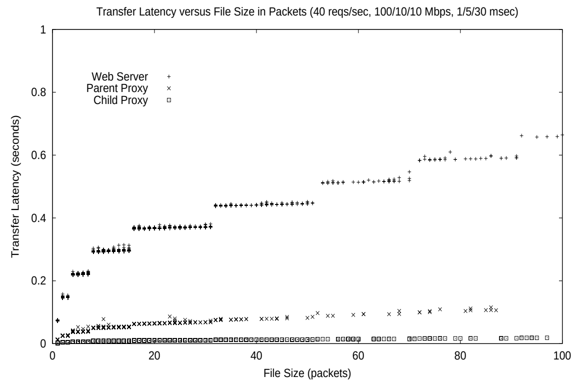
The most obvious difference between Figures 7(a) and (b) is the “split” between the Child Proxy and Parent Proxy transfers. The Child Proxy only handles small transfers (at most 16 packets), while the Parent Proxy only handles larger ones. Transfers from the Web server can be of any size, since the Child Proxy and the Parent Proxy each cache only a subset of the total eligible documents.



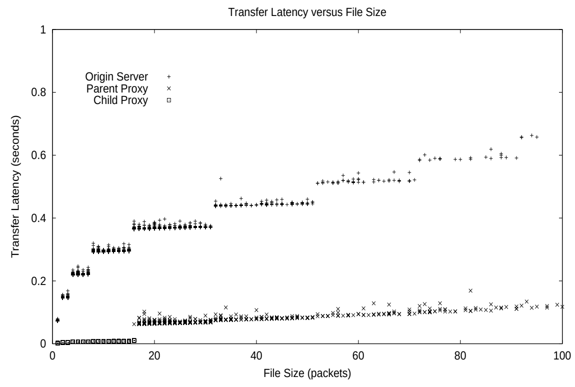
(a) Default Policy, $C_1 = 10$ Mbps



(b) **Size** Threshold Policy, $C_1 = 10$ Mbps



(c) Default Policy, $C_1 =$ **100** Mbps



(d) **Size** Threshold Policy, $C_1 =$ **100** Mbps

Figure 7. Effect of Cache Management Policy

Table 5. Simulation Results for Size-Based Thresholding Experiment ($\lambda = 40$ requests/sec)

Item	Default	Threshold	Default	Threshold
Link Capacity C_1 (Mbps)	10	10	100	100
Total HTTP Transfers	5,000	5,000	5,000	5,000
Mean Transfer Latency (sec)	**0.156**	**0.150**	**0.144**	**0.148**
Web Server Transfers	2,252	2,187	2,252	2,187
Mean Web Server Transfer Latency (sec)	0.306	0.303	0.295	0.301
Parent Proxy Transfers	758	723	758	723
Mean Parent Proxy Transfer Latency (sec)	**0.058**	**0.096**	**0.052**	**0.092**
Child Proxy Transfers	1,990	2,090	1,990	2,090
Mean Child Proxy Transfer Latency (sec)	**0.025**	**0.009**	**0.009**	**0.006**

The second observation of note in Figure 7(b) is the reduced impact of network congestion on the transfer latencies, compared to Figure 7(a). While the number of transfers from the Child Proxy is about the same as in Figure 7(a), the number of packets is much lower, since the average transfer size is smaller. As noted earlier, this changes the packet arrival process at the bottleneck link L_1. Child Proxy transfers send fewer packets, with smaller congestion window sizes, making the packet arrivals less bursty. As a result, there is less queueing and packet loss, and these transfers complete sooner, reducing contention for network resources. Similarly, the "large" transfers that are "pushed" up to the Parent Proxy generate their packets over a longer connection duration, alleviating congestion on the bottleneck link L_1. However, congestion can still be a problem at link L_2 or L_3.

Whether the size-based thresholding policy is a "win" or not depends on the decrease in Child Proxy transfer latency versus the increase in Parent Proxy transfer latency. (The overall impact on Web Server transfer latency is negligible.) Table 5 summarizes these results. In our experiments, we have found that when $C_1 = 10$ Mbps, the size-based thresholding policy provides *marginally* better mean response time (0.150 versus 0.156) than the default policy when the request rate is 40 requests/sec (see Table 5), and *much better* mean response time than the default policy when the request rate is 80 requests/sec (0.163 versus 0.214, not shown here). However, when $C_1 = 100$ Mbps, the size-based thresholding policy is *marginally worse* than the default policy (0.148 versus 0.144, in Table 5) when the request rate is 40 requests/sec, and *much worse* than the default policy when the request rate is 80 requests/sec (0.160 versus 0.145, not shown here). In other words, size-based thresholding (with small objects close to clients) makes sense if and only if the client access link is a bottleneck.

These results confirm the speculation in [11] that the performance of size-based thresholding is highly dependent on network-level effects, such as TCP behaviours, RTTs, and network congestion. Our packet-level simulation study has illustrated exactly these effects.

CONCLUSIONS

This paper has used a combination of application-level Web caching simulation and packet-level network simulation to illustrate the impacts of network-level effects on user-perceived Web performance. In particular, the paper considers a simple two-level Web proxy caching hierarchy, and studies the transfer latencies for Web document downloads when the characteristics of the network topology and Web workload are varied. The main observation is that link capacity, round-trip time, network congestion, and TCP behaviours all have a significant influence on the user-level response time. The simulation results also highlight the subtleties of the relationship between cache hit ratio and user-perceived Web performance.

Ongoing work is extending the simulations to much longer workloads on larger and more interesting network topologies. The impacts of different variations of the TCP protocol (e.g., SACK TCP, CATNIP TCP [26]) are also being studied.

ACKNOWLEDGMENTS

Financial support for this research was provided by iCORE (Informatics Circle of Research Excellence) in the Province of Alberta, the Canada Foundation for Innovation (CFI), and by the Natural Sciences and Engineering Research Council (NSERC) of Canada. The authors thank the anonymous SPECTS reviewers for their feedback on an earlier version of this paper.

References

[1] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing", *Proceedings of IEEE INFOCOM*, pp. 1157-1165, Tel Aviv, Israel, March 2000.

[2] M. Allman, C. Hayes, and S. Ostermann, "An Evaluation of TCP with Larger Initial Windows", *ACM*

Computer Communication Review, Vol. 28, No. 3, pp. 41-52, July 1998.

[3] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[4] M. Arlitt and C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 631-645, October 1997.

[5] M. Arlitt and C. Williamson, "Trace-Driven Simulation of Document Caching Strategies for Internet Web Servers", *Simulation Journal*, Vol. 68, No. 1, pp. 23-33, January 1997.

[6] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz, "TCP Behavior of a Busy Internet Server: Analysis and Improvements", *Proceedings of IEEE INFOCOM*, pp. 252-262, San Francisco, CA, March 1998.

[7] P. Barford and M. Crovella, "Critical Path Analysis of TCP Transactions", *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, pp. 127-138, September 2000.

[8] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", *Proceedings of IEEE INFOCOM*, New York, NY, pp. 126-134, March 1999.

[9] L. Breslau *et al.*, "Advances in Network Simulation", *IEEE Computer*, Vol. 33, No. 5, pp. 59-67, May 2000.

[10] M. Busari and C. Williamson, "On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics", *Proceedings of IEEE INFOCOM*, pp. 1225-1234, Anchorage, AL, April 2001.

[11] M. Busari and C. Williamson, "Simulation Evaluation of a Heterogeneous Web Proxy Caching Hierarchy", *Proceedings of IEEE MASCOTS*, pp. 379-388, Cincinnati, OH, August 2001.

[12] M. Busari and C. Williamson, "ProWGen: A Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches", *Computer Networks*, Vol. 38, No. 6, pp. 779-794, June 2002.

[13] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich, "Web Proxy Caching: The Devil is in the Details", *ACM Performance Evaluation Review*, Vol. 26, No. 1, pp. 11-15, December 1998.

[14] H. Che, Z. Wang, and Y. Tung, "Analysis and Design of Hierarchical Web Caching Systems", *Proceedings of IEEE INFOCOM*, pp. 1416-1424, Anchorage, AL, April 2001.

[15] R. Doyle, J. Chase, S. Gadde, and A. Vahdat, "The Trickle-Down Effect: Web Caching and Server Request Distribution", *Proceedings of the Web Caching and Content Delivery Workshop*, Boston, MA, June 2001.

[16] L. Eggert, J. Heidemann, and J. Touch, "Effects of Ensemble-TCP", *ACM Computer Communication Review*, Vol. 30, No. 1, pp. 15-29, January 2000.

[17] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich, "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments", *Proceedings of IEEE INFOCOM*, pp. 107-116, New York, NY, April 1999.

[18] A. Feldmann, J. Rexford, and R. Caceres, "Efficient Policies for Carrying Web Traffic Over Flow-Switched Networks", *ACM/IEEE Transactions on Networking*, Vol. 6, No. 6, pp. 673-685, December 1998.

[19] R. Jain, *The Art of Computer Systems Performance Analysis*, Wiley, 1991.

[20] J. Ke and C. Williamson, "Towards a Rate-Based TCP Protocol for the Web", *Proceedings of IEEE MASCOTS*, San Francisco, CA, pp. 36-45, October 2000.

[21] A. Mahanti, C. Williamson, and D. Eager, "Traffic Analysis of a Web Proxy Caching Hierarchy", *IEEE Network*, Vol. 14, No. 3, pp. 16-23, May/June 2000.

[22] N. Markatchev and C. Williamson, "WebTraff: A GUI for Web Proxy Cache Workload Modeling and Analysis", *Proceedings of IEEE MASCOTS*, pp. 356-363, Fort Worth, TX, October 2002.

[23] J. Mogul, "The Case for Persistent Connection HTTP", *Proceedings of ACM SIGCOMM*, pp. 299-313, Boston, MA, August 1995.

[24] V. Padmanabhan and R. Katz, "TCP Fast Start: A Technique for Speeding up Web Transfers", *Proceedings of IEEE GLOBECOM'98 Internet Mini-Conference*, Sydney, Australia, pp. 41-46, November 1998.

[25] C. Williamson, "On Filter Effects in Web Caching Hierarchies", *ACM Transactions on Internet Technology*, Vol. 2, No. 1, pp. 47-77, February 2002.

[26] C. Williamson and Q. Wu, "A Case for Context-Aware TCP/IP", *ACM Performance Evaluation Review*, Vol. 29, No. 4, pp. 11-23, March 2002.