Using Proxies to Enhance TCP Performance over Hybrid Fiber Coaxial Networks

Reuven Cohen
HP Israel Science Center
Technion City, Haifa 32000, Israel

and

Srinivas Ramanathan
Hewlett-Packard Laboratories,
1501 Page Mill Road, M/S 1U-17
Palo Alto, CA 94304, U.S.A.

Abstract

Using cable modems that operate at several hundred times the speed of conventional telephone modems, many cable operators are beginning to offer World Wide Web access and other data services to residential subscribers. Initial experiences indicate that real-world Hybrid Fiber Coaxial (HFC) networks are susceptible to a variety of radio-frequency impairments that significantly reduce the benefits of using high-speed cable modems. The effects of packet losses in the access network are particularly accentuated during subscriber accesses to remote servers on the Internet. The longer round-trip times in such accesses together with the high packet loss rate result in dramatic degradations in performance perceived by subscribers. This paper shows that by using proxy servers to handle all remote accesses from an HFC access network, the performance of remote accesses can be significantly enhanced even in cases when the proxy servers do not function as data caches. By handling packet losses that occur in the HFC network locally, at low latencies and without the remote server even being aware of the loss, a proxy server enables faster recovery from packet losses. Most importantly, since it controls data transmissions over the local HFC network, the proxy server's TCP implementation can be optimized for the loss characteristics of the HFC access network, enabling a significant increase in performance when the access network is lossy.

Keywords: Web proxies, TCP performance, Hybrid Fiber Coaxial networks, Residential broadband data services.

1 Introduction

The evolution of the World Wide Web in recent years has dramatically altered the nature and frequency of Internet accesses. It is estimated that in the last two years alone, there has been a three-fold increase in the number of Internet hosts. During this period, the Web itself has seen a twenty-fold growth in the number of servers. Currently, there are over 230,000 Web servers on the Internet, offering content to over 12 million hosts [1]. Network traffic statistics also indicate a significant growth during this period. The Hypertext Transport Protocol (HTTP) and the File Transfer Protocol (FTP) have emerged as the predominant protocols in use on the Internet, together accounting for almost 50% of the traffic on the NSFNET backbone [2]. From being an experimental research vehicle, the Internet has evolved as the basis for the Information Super Highway, over which on-line information dissemination, electronic commerce, multimedia conferencing, and a variety of other advanced services are available to users. To cash in on the popularity of the Web and the Internet, many cable operators are beginning to offer a variety of data services including E-mail, News, and Web access to residential subscribers over their Hybrid Fiber Coaxial (HFC) access networks. Using cable modems operating at speeds ranging from hundred to thousand times the speed of conventional telephone modems, these services are expected to enable access from the home at broadband speeds [3].

Initial experiences with these data services indicate that many times, the projected increase in access bandwidth does not translate into a corresponding increase in the throughput perceived by subscribers. This discrepancy is attributable to the susceptibility of real-world cable networks to a variety of radio-frequency impairments, such as ingress noise, multi-path fading, cross modulation, etc, referred to in this paper as *media errors* [4]. The Transmission Control Protocol (TCP) that is predominantly used by data applications has been optimized for use in the Internet, where packet losses mainly occur due to congestion. When it encounters packet losses caused by media errors, TCP reacts in the same way as it does when it experiences packet losses due to network congestion: it dramatically lowers its packet transmission rate, and thereby causes a significant reduction in the throughput perceived by subscribers [5]. Since the time taken by TCP to recover from losses and to begin operating at its maximum rate is dependent on the round-trip delay between the servers and clients, the performance degradation is accentuated during remote accesses over long distances, from the HFC access network to the Internet and other external networks.

This paper focusses on methods for increasing the performance of remote accesses initiated by subscribers over an HFC access network. Since a majority of remote accesses are expected to be HTTP and FTP requests to the World Wide Web, we explore the advantages of using a proxy server in the network headend. Using simulations of a typical HFC network, we show that even when it does not serve as a data cache for remote content, the proxy server can offer significant performance improvement for remote accesses, especially during times when the HFC network is prone to losses. By splitting the remote connection into two distinct connections that operate in parallel and by reducing the round-trip latency for each connection, the proxy server speeds up end-to-end performance. By handling packet losses that occur in the HFC network locally, at low latencies and without the remote server even being aware of the loss, the proxy enables faster recovery from packet losses. Most importantly, since it controls data transmissions over the local connection, the proxy server's TCP implementation can be optimized for the loss characteristics of the HFC access network, enabling a significant increase in performance when the access network is lossy. Simulations demonstrate that a proxy server tailored for the HFC network yields a several fold performance increase under certain loss conditions. The exact

performance gain depends on the loss rates and round-trip delays in the local and remote network segments, the data transfer size, the buffering capacities of the proxy server, the cable modems, and subscribers' PCs, and on the processing capability and degree of loading of the proxy server.

The rest of this paper is organized as follows. Section 2 introduces the typical architecture for broadband data services and discusses network performance issues in HFC networks. Using simulations, Section 3 highlights the advantages of proxies for speeding subscriber accesses from the HFC network to other external networks including the Internet. Optimizations that can be implemented at proxies to yield further speed-up are also discussed in this section. Section 4 evaluates the effectiveness of proxies for different data transfer sizes, locations of remote servers, and different loss conditions on the Internet. Finally, Section 5 summarizes the contributions of this work and outlines areas for future research.

2 Broadband Data Services over HFC Networks

2.1 System Architecture

Figure 1 depicts a typical architecture for supporting data services over HFC networks. Subscribers connect to the HFC network using cable modems (CMs) that implement sophisticated modulation-demodulation circuitry to transmit and receive analog modulated digital signals over the HFC network. Bridging equipment, referred to as the *Signal Conversion Systems (SCS),* performs the modulation-demodulation functions for transmission and reception of signals at the network headend. In order to co-exist with analog television broadcasts to the home, separate frequency channels are used in the HFC network for communications between the SCS and CMs. Because of the asymmetric nature of Web accesses in the Internet, the SCS-CM communication links are being designed with higher transfer speeds (4-30 Mbps) on the downstream channels that are used for communications from the headend to the home, than on the upstream channels (0.5-10 Mbps), that are used for communications from the home to the headend. To mediate accesses by dif-

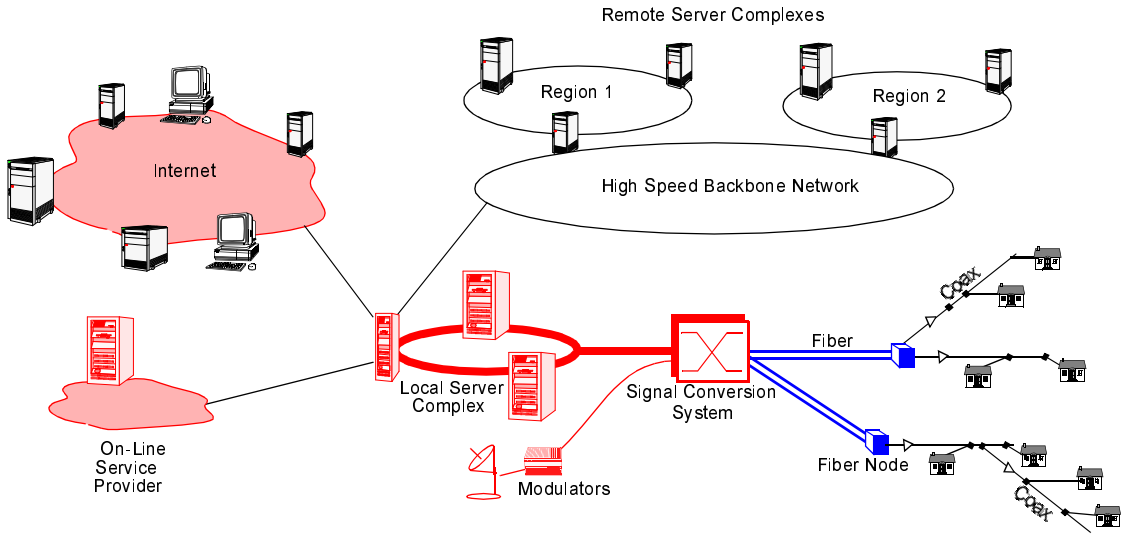


Figure 1. Delivery of broadband data services via interconnection of high speed networks

ferent CMs to the shared upstream channels, various media access control (MAC) protocols specific to the cable infrastructure have been proposed [3,6].

From their home PCs that are connected via 10BaseT interfaces to the CMs, subscribers can access a variety of data applications including E-mail, News, and the Web. These services are provisioned using servers co-located with analog television broadcast equipment in the headend. The HFC network also provides subscribers connectivity to the Internet and to external service providers such as America On Line. The local server complexes in different metropolitan regions served by the same operator may also be interconnected to facilitate content sharing and caching.

To integrate the access network with the Internet, thereby making the various Internet applications available to subscribers, IP is the network layer protocol of choice for communication between subscribers' PCs, the local server complex, and the other external networks.

2.2 Performance of Broadband Data Services

One of the major factors governing the subscriber acceptance and popularity of broadband data services is their capability to deliver on the promise of superior performance compared to more conventional ways of accessing data services. A key problem in meeting subscribers' performance expectation is the susceptibility of HFC networks to various forms of radio-frequency (RF) impairment - from unterminated connections in the home, cracked cables, misaligned amplifiers, misconfigured power level settings, etc. The low frequencies that are used by the upstream channels for communication from the home to the headend are especially vulnerable to noise attacks [4]. The increased bit error rates on the noisy channels manifest as increased packet loss rates, as seen by TCP applications.

Our initial experiences with monitoring data services over HFC networks indicate that packet losses of 1-10% are common, especially on the upstream channels [7]. The precise loss rate may vary from one network to another, and even from one neighborhood to another in the same network, depending on many factors including nature of wiring both in the network and inside subscribers' homes as well as the nature of external noise sources. Features of the SCS and CMs such as the use of Forward Error Correction (FEC) for SCS-CM communications, the ability for the SCS to monitor and change channels when it notices an increase in errors on the channel, the MAC protocol in use and its vulnerability to packet losses, etc., can reduce the impact of RF impairments on performance of the data services [5].

2.2.1 Performance of Local Accesses

In [5], the performance of TCP applications operating over lossy HFC access networks was analyzed. This analysis indicates that TCP applications are much more susceptible to loss of data packets than to loss of acknowledgement (ACK) packets. Whereas a relatively small loss rate of 1% of the data packets causes almost a dramatic 50% reduction in TCP throughput for data transfers of 300KB and more, even a 10% loss of ACKs reduces throughput by only 10%. The relatively small degradation in performance with ACK loss is attributable to the use of cumulative acknowledgements in TCP, whereby every ACK transmitted by the receiver of a connection acknowledges all the data it has received prior to the transmission of the ACK. Consequently, when an ACK is lost, the ACK for a subsequent packet enables a TCP transmitter to recover from the loss without retransmissions. On the other hand, when a data packet is

lost, the TCP transmitter has to retransmit the lost packet. More importantly, since TCP is not able to distinguish packet loss caused by media errors from packet loss induced by network overload, whenever it notices packet loss, the transmitter invokes TCP's congestion control mechanisms to slow down its transmission rate. There are two basic mechanisms that the transmitter employs to recover from loss [8]:

- *Fast retransmit and recovery:* This mechanism enables a TCP transmitter to detect and recover from isolated packet losses. When one packet out of a group of packets in a transmission window is lost, the TCP receiver notices a gap in the packet sequence and issues a duplicate ACK (dupACK) indicating the packet that it next expects to receive in sequence. Upon receiving three dupACKs for a packet, the transmitter immediately retransmits the packet. When an ACK for the retransmitted packet is received at a later time, attributing the packet loss to a possible over utilization of the network, the transmitter slows down its transmission rate to half its original value. The transmitter then enters a congestion avoidance phase in which it slowly increases its transmission rate by one packet for every round-trip time.

  The rate of window growth may be curtailed if the receivers implement a delayed ACK strategy. As per this strategy, the receiver holds back the transmission of an ACK for a received packet until a subsequent packet is received, at which time it transmits a cumulative ACK to acknowledge both the received packets. RFC 1122 which describes delayed ACK indicates that the receiver must transmit one ACK whenever it receives data equivalent to two maximum-sized packets [9]. While using delayed ACKs, the maximum period for which the receiver can wait for a subsequent packet is limited to about 200ms, this value being determined by a delayed ACK timer that TCP maintains.

- *Time-outs and Slow-Start:* When multiple packet losses occur or when the transmission window is less than four packets and an isolated packet loss occurs, the transmitter may not receive the three dupACKs necessary for it to trigger the fast retransmit strategy. To recover from such cases, the TCP transmitter uses a retransmission timer to track the maximum round-trip time of packets transmitted over the connection. At any time, the transmitter tracks one of the packets in the transmission window and uses the measured round-trip time for this packet to update an estimate that it maintains for the maximum round-trip time over the TCP connection. When fast retransmit fails to discover packet losses, the transmitter ends up waiting for the retransmission timer to indicate that the maximum round-trip time has been exceeded for a packet, implying that one or more packets have probably been lost.

  Following the long waiting period, referred to as a time-out, assuming the packet losses to be caused by a large change in available network bandwidth, the transmitter enters a slow-start phase by setting its transmission window to one packet. The transmitter then retransmits the lost packets as permitted by the transmission window. The receipt of ACKs during slow-start is assumed to be a signal that the network can support more transmissions, and so the transmitter increases its transmission window exponentially during each round-trip time. This increase continues until the transmission window reaches half the value it had when the time-out occurred. Beyond this point, the transmitter shifts to the congestion avoidance phase, in which the window is increased linearly during each round-trip [8].

  The use of a 500ms retransmission timer in most commercial TCP implementations limits the accuracy of the TCP transmitter's estimation of the round-trip time for a connection. Furthermore, since TCP's round-trip estimate is based on the sum of the mean and four-times the deviation of the measured round-trip times, the minimum TCP time-out

period is of the order of 2-3 seconds [8]. The long time-out periods and the increase in frequency of time-outs with increase in packet loss rates are the main reasons for the dramatic reduction in throughput in lossy HFC networks.

For increasing the performance of subscriber accesses to the local server complex of a broadband data system, two simple modifications to TCP implementations at the local servers are proposed in [5] :

- *Using a finer granularity retransmission timer*: By enabling the TCP transmitter to estimate round-trip times more accurately, the finer granularity timer reduces the duration of time-outs. Using a 200ms timer, which matches the granularity of the timer used for delayed ACKs, increases TCP's reactivity without greatly increasing its implementation overhead.
- *Initiating fast retransmit after the first duplicate ACK is received*: Since in-sequence delivery of packets is typically guaranteed over the HFC network and all the way to the local server complex, the receipt of the first duplicate ACK at a local server is a sufficient signal that a packet has been lost in the downstream direction. By triggering fast retransmit soon after the first duplicate ACK is received at a local server, the frequency of TCP time-outs can be significantly reduced.

Whereas the first modification reduces the duration of each TCP time-out, the second modification reduces the frequency of time-outs. Together, these modifications have been shown to yield more than two-fold improvement in performance for local accesses.

2.2.2 Performance of Remote Accesses

This paper focusses on methods for enhancing the performance of subscriber accesses from the HFC network to the Internet (and other external networks). The longer round-trip times in such remote accesses only serve to accentuate TCP's poor performance during times when the HFC network is prone to media errors. This is because the rate of increase of TCP's transmission window during the fast retransmit and slow-start stages is inversely proportional to the round-trip time. Hence, longer the round-trip time, slower the recovery from packet loss.

The modifications described above for local accesses cannot be directly applied for remote accesses for several reasons. Firstly, the Internet has different routing characteristics than the HFC access network. In an HFC network, there is typically a single, static route between a subscriber's PC and the local server complex, whereas in the Internet packets transmitted over the same connection can be routed over different paths and may not arrive in sequence at the receiver. Since out-of-sequence delivery is common in the Internet, modifying the fast retransmit implementation at the remote server to retransmit packets after the first dupACK is received by the server can result in unnecessary retransmissions and reduction of the server's transmission window. A further problem with applying the TCP modifications described in the previous section in the context of the remote accesses is the need to modify the large number of servers on the Internet. In fact, such modifications may not even be possible since the servers on the Internet are autonomously owned and managed.

A common approach adopted in LANs for improving the performance of TCP connections is to increase the socket buffer size. However, it has been demonstrated in [5] that in HFC networks, arbitrarily increasing the TCP socket buffer

size without considering the buffer capacity of the CMs and the SCSs can result in buffer overruns at these devices, and consequently resulting in significantly lower throughput. Because of the cost-sensitive and competitive CM market, it is expected that the CM buffer capacity will limit the TCP socket buffer size setting.

For enhancing performance of remote accesses, we draw upon the idea of "split connections" first proposed in the context of interconnections between wireless networks and wired networks [10]. To handle packet losses that are caused in wireless networks by the mobility of the end hosts, [10] proposes that wireless base stations that interconnect the wired and wireless networks transparently split each end-to-end transport connection into two connections, one that operates over the wired network and the other over the lossy wireless network. The TCP implementations at the wireless base stations are modified so that upon noticing link layer indications of mobility, slow-start is triggered early, without having to wait for long time-out periods. The effectiveness of the split-connection approach in localizing errors in wireless networks and the consequent speed up of the end-to-end connection is illustrated in [11]. The main drawback of this approach is that instead of using normal TCP connections, applications must establish special, Indirect-TCP (I-TCP) connections with the wireless base stations. Hence, implementing this approach requires changes to existing application software in all the hosts on the wireless network. Another problem is that since the end-to-end connection is split transparently, without the server on the wired network being aware of the split, this approach does not preserve end-to-end TCP semantics. More specifically, an ACK received by the server is no longer an unequivocal indication that the corresponding packet has been delivered or will be delivered to the intended receiver.

Focussing on HTTP and FTP traffic, which constitute a majority of the traffic from an HFC network to the Internet, we propose to improve the performance of remote accesses by using application-level proxy servers to split the end-to-end connection between a subscriber's PC and a remote server into two TCP connections: a *local connection* between the subscriber PC and the proxy server over the HFC network, and a *remote connection* between the proxy server and the remote server over the external network(s). This approach is especially attractive because most Web browsers already support the use of proxy servers, and hence, separate local and remote connections can be established without requiring any change in the applications. Moreover, since proxy servers are already being used for Web accesses for other tasks, such as caching, security, and protocol conversion, the current Web access model itself does not preserve the end-to-end semantics. Figure 2 illustrates the protocol interactions for proxied connections.

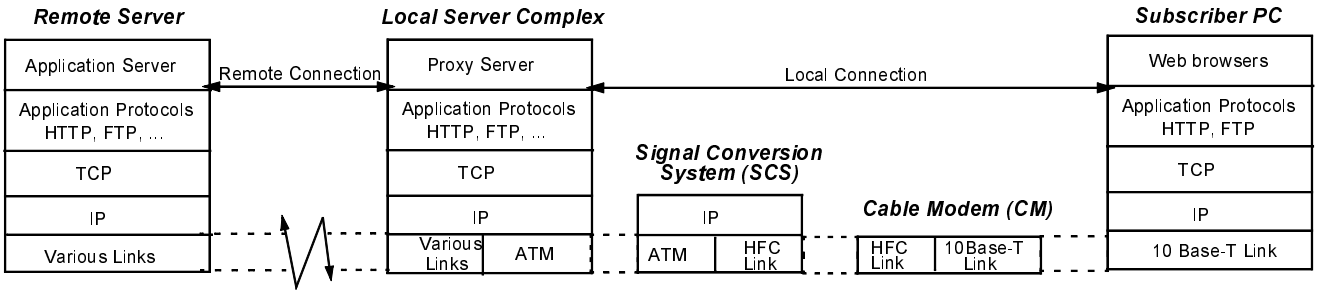


Figure 2. Interactions across the different protocol layers for supporting proxied connections. The proxy server splits the end-to-end connection into a remote connection and a local connection.

In the following sections, we study the performance of proxy servers under different HFC network conditions and propose ways of configuring the proxy servers to ensure maximum end-to-end performance. Our analysis demonstrates that proxy servers can offer significant performance improvements in HFC networks even when they do not perform any caching functions.

3 Evaluating the Effectiveness of Proxies in HFC Networks

3.1 Network Model

To study the effectiveness of proxies in HFC networks, we have used the *ns* simulator from LBL to model a typical HFC network and its links to the Internet [12]. Figure 3 depicts the network configuration used in the simulations. The downstream and upstream channels on the HFC network operate at 25 Mbps and 3 Mbps, respectively. A 100Mbps ATM link interconnects the SCS and the local server complex. The local server complex is connected to the Internet via a T-3 link operating at 45 Mbps. Based on experiments in real-world deployments of data services over HFC networks [7], the round-trip time of packets transmitted between the local server complex and a subscriber's home is set to 20ms. The round-trip time to access a remote server over the Internet is assumed to be 100ms.

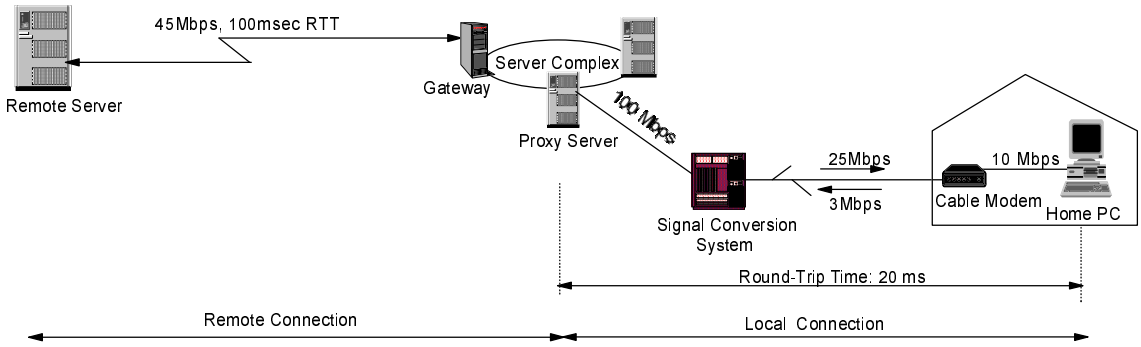


Figure 3. Configuration of the simulated HFC network

The cable modems are assumed to have 10KByte (KB) buffers. Assuming that on an average three simultaneous TCP connections are supported via the cable modem, the TCP socket buffer size at the PC is restricted to 8KB in order to avoid buffer overruns at the cable modem during access to the local servers. This is based on an analytical method described in [5] for deriving the optimal TCP socket buffer size based on the cable modem buffer size, the HFC network bandwidths, and the number of simultaneous connections.

Since we are interested in performance under loss caused by media errors rather than under congestion, the precise contention resolution and network access algorithms of the HFC upstream link protocol are not modeled in the simulator. Furthermore, in the simulations, the number of simultaneous connections over the HFC network are controlled so as to avoid network overload. In the absence of precise models for media errors that happen over HFC networks, loss of TCP packets and acknowledgements are modeled using poisson distributions.

The network traffic is assumed to be HTTP and FTP access to the Internet. A wide range of data transfer sizes are studied to characterize the effectiveness of proxies for different content (data and image intensive) and protocols (e.g., HTTP 1.1 [13], which proposes to use a single persistent connection for access to all the components of a Web page). All TCP receivers and transmitters on the network are assumed to implement TCP Reno [8]. In keeping with most commercial TCP implementations, the receivers are assumed to implement delayed ACKs [9]. The TCP data packets and ACKs are assumed to be 536 bytes and 40 bytes in length, respectively.

3.2 Proxy Models

To study the performance advantages that a proxy server offers even when it does not function as a data cache, caching algorithms for the proxy server are not modeled in the simulations. Hence, for each request it receives, the proxy establishes a local connection and a remote connection. Two models for the operation of the proxy are used in the simulations (see Figure 4):

- *Store-and-forward proxy*: In this model, the proxy server uses intermediate storage (e.g., disk) to permit the remote connection to operate asynchronously with respect to the local connection. For doing so, the proxy server first temporarily stores all the data received over the remote connection. When permitted by the transfer rates over the local connection, the data is then retrieved from intermediate storage and transmitted over the local connection.
- *Buffer-and-forward proxy*: In this second model, the proxy directly queues all the data received over the remote connection for transmission over the local connection without using intermediate storage. By doing so, the proxy forces the remote connection to operate in synchrony with the local connection.

Many proxy server implementations have adopted the buffer-and-forward model due to its simpler design. A few of the newer designs are adopting the more complicated store-and-forward model. We consider the two proxy models to illustrate some of the advantages of the store-and-forward model in lossy HFC networks and demonstrate that these advantages can be obtained in the buffer-and-forward model as well, by tuning their TCP implementations to handle packet losses better. In the following sections, we evaluate the relative effectiveness of both of these models under different HFC network conditions. In the simulations, disk storage and disk access times are assumed not to be a bottlenecks. Furthermore, the proxy server is assumed to have sufficient processing and communication capacity so as to not be the bottleneck in the system. Scalability of proxy servers is an important research problem that is outside the scope of this

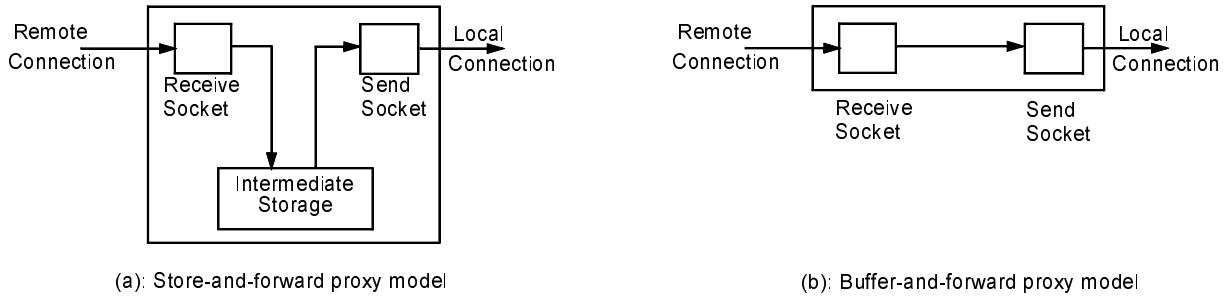


Figure 4. Different models of proxy server operation: (a) a store-and-forward model, (b) a buffer-and-forward model.

paper. Because the proxy server is assumed to not be a cache, cache lookup times are not considered in the simulations. When a proxy server is used as a cache, the contribution of cache lookup times to the overall data transfer time must be considered while evaluating the benefits of proxy servers.

3.3 The Advantage of Split Connections

Before evaluating the effectiveness of a proxy server during times when the HFC network is prone to errors, we first consider the case when the HFC network is error-free. To understand the effectiveness of proxies, we first compare the end-to-end throughput achieved in the case when a proxy is used and the case when a direct connection is established between the subscriber's PC and a remote server. Three different scenarios of subscriber access are considered by varying the round-trip time over the remote connection. In all of the scenarios, the socket buffer sizes used for the local and the remote connections are assumed to be 8KB. Figure 5 illustrates that even when the HFC network is lossless, the proxy server offers a significant improvement in performance when the round-trip time over the remote connection is comparable to that for the local connection. For example, when the round-trip time over the remote connection is 100ms, the proxy yields a 25% improvement in throughput for data transfers of 500KB and above. Both the proxy models yield similar performance since packet loss does not occur.

To understand the reason for the performance improvement when using a proxy, observe that by splitting the connection between the remote server and the PC into two separate connections, the proxy server permits data transfer over the two connections to be pipelined. In doing so, the proxy reduces the effective round-trip time for each connection. Specifically, packets transmitted over the remote connection are acknowledged by the proxy itself, while ACKs transmitted by the PC over the local connection need only reach the proxy before subsequent packets are transmitted to the PC. Since the throughput achieved over a connection for a constant socket buffer size is directly related to the round-trip time, the throughput achieved on the local and remote connections are both greater than the throughput achieved over

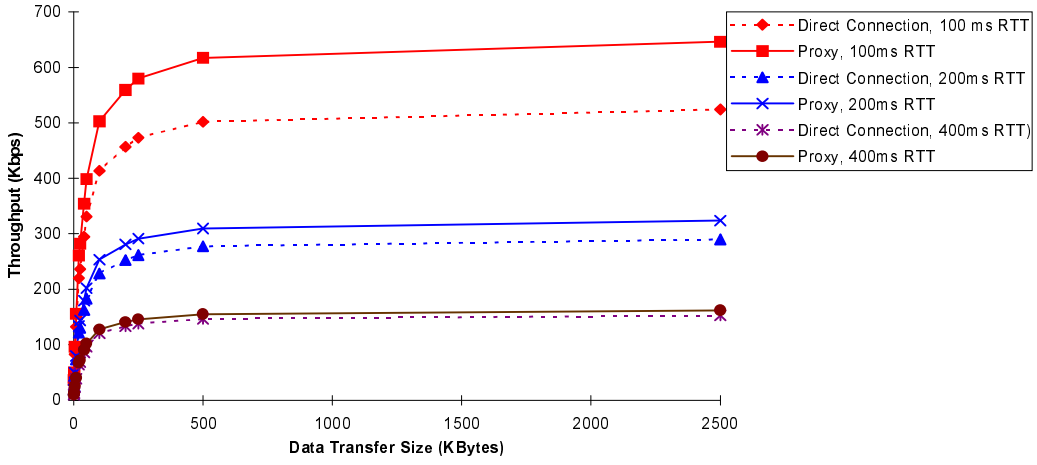


Figure 5. Comparison of the end-to-end performance obtained when using a proxy server with that obtained using a direct connection to transfer data from a remote server to the subscriber's PC. Three scenarios with different round-trip times for the connection between the proxy server and the remote server are considered. In all cases, the round-trip time for the connection between the subscriber's PC and the proxy server is 20ms.

the direct connection. Hence, the end-to-end throughput in the proxy approach, which is the minimum of the throughput values for the local and remote connections, is higher than that achieved when using a direct connection.

Because of the smaller round-trip time on the local connection, the absence of packet losses in either connection, and the use of the same socket buffer size for both connections, the end-to-end throughput in the proxy approach is limited by the throughput achieved on the remote connection. Therefore, the magnitude of the speed-up achieved by using a proxy is directly related to the ratio of the round trip time for the direct connection and the round-trip time for the remote connection. Assuming the round-trip time for the local connection to be fixed, the speed-up is inversely related to the round-trip time for the remote connection, i.e., the smaller the round-trip time, the greater the speed-up (see Figure 5).

Figure 5 also depicts that the performance improvement is greater for larger transfers than for smaller transfers. This is because during the initial stages of the local and remote connections, their transmission windows are so small that there is little overlap between data transfers over the two connections. As the data transfer proceeds, after several round-trips, the transmission windows for the local and remote connections grow and become large enough to enable a significant overlap in the data transmissions over the two connections. Consequently, larger data transfers that better exploit the overlap in data transmission over the two connections yield a higher improvement in throughput.

In the absence of any packet loss, since the throughput achieved over the remote connection is not greater than that achieved over the local connection (because of the longer round-trip time over the remote connection), packets are never queued at the proxy server for transmission over the local connection. Hence, the two models behave identically, yielding the same end-to-end throughput.

3.4 Avoiding the Effects of Delayed ACK

In [5], the adverse effect that the implementation of delayed ACK in the TCP stack of subscribers' PCs can have on performance was noted, even in cases when the access network is neither prone to errors nor over loaded. This performance degradation is attributable to the interaction between the slow-start strategy that is used by the TCP transmitters and the delayed ACK strategy supported in TCP receivers. For HTTP transfers from a local server, slow-start implementation at the local server restricts the server's transmission window to two maximum sized TCP data packets (the ACK transmitted during TCP connection establishment increases the server's transmission window to 2 packets). Since many commercial HTTP server implementations transmit the HTTP response header separate from the data and since the header is typically much smaller than a maximum sized TCP data packet, the server can subsequently transmit only one data packet following the HTTP response header before it is forced to wait for an ACK from the subscriber's PC (see Figure 6). Since the data received by the subscriber PC is less than two maximum sized TCP data packets, the subscriber PC delays its ACK, waiting for up to 200ms before acknowledging the received packets. This long initial wait time significantly impacts performance for most common HTTP transfers.

Increasing the minimum TCP transmission window by one packet improves performance significantly. With this modification, data transfers under 150KB proceed almost twice as fast as before. While such a change could benefit all transfers (whether from remote or local servers), implementing such a change is only viable under local control. Hence, this

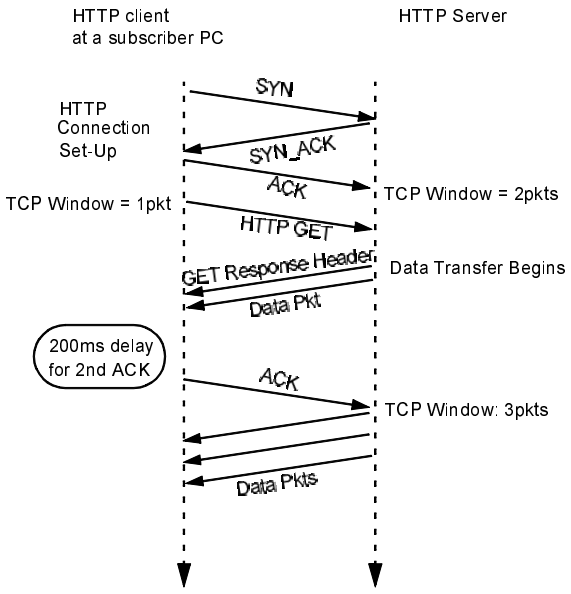


Figure 6. Illustration of the effect of delayed ACK implementation in TCP stacks of subscriber PCs on performance o HTTP data transfers. The initial delay of up to 200ms for the second ACK from the subscriber PC reduces throughput by over 50% for data transfers sizes of up to 150 KB

same approach cannot be adopted to speed-up remote accesses because the remote server being accessed, which controls the size of the transmission window, may not be under the data service operator's control. If a direct connection is used for communication between the subscriber's PC and the remote server, the deleterious effects of delayed ACK cannot be avoided. When a proxy server is used for remote accessesup to, there are two possible approaches to counteract delayed ACKs:

- *Avoiding delayed ACK implementation at the proxy server as the receiver of the remote connection:* Delayed ACK primarily benefits symmetric applications in which the ACKs can be piggybacked with data packets, and interactive applications, such as rlogin and telnet, in which a cumulative ACK can be generated for several small packets. For a proxy server that is predominantly used for HTTP and FTP accesses, the benefit of delayed ACK is minimal. By avoiding delayed ACK implementation at the proxy server (at least in the initial stages of a TCP connection), we can ensure that the remote connection does not experience the initial 200ms delay. Figure 7 illustrates the performance gain that can be obtained by avoiding delayed ACK implementation at the proxy server. The significant performance improvement that this modification to the TCP stack of the proxy server offers far out-weighs the small extra consumption of bandwidth over the remote network that results from more frequent transmission of ACKs from the proxy server.
- *Increasing the initial TCP window used by the proxy server as the transmitter of the local connection by one maximum sized data packet:* This modification ensures that the local connection does not experience the long initial waiting time because of delayed ACK implementation at the subscriber's PC.

The above modifications are applicable to both the proxy models described in Section 3.2.

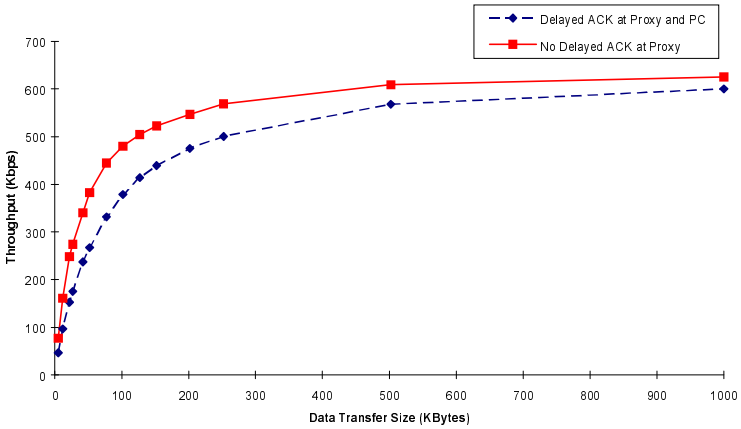


Figure 7. Improving the throughput obtained during remote accesses by avoiding delayed ACK implementation at the proxy server. The round-trip time between the subscriber's PC and the proxy server is 20ms, and that between the proxy server and the remote server is 100ms.

Observe from Figure 7 that the first modification above yields up to 30% increase in performance for data transfers of 150KB and less. The precise performance gain is dependent on the round-trip time over the remote connection. The larger the round-trip latency, the smaller the performance gain because the contribution of the initial waiting time to the total data transfer time decreases with an increase in round-trip latency. Since the contribution of the initial waiting time also drops in magnitude for longer data transfers, the performance gain from avoiding delayed ACK at the proxy server also drops with increase in data transfer size. The limit on the throughput achieved in Figure 7 is imposed by the socket buffer setting at the PC (8KB) and the round-trip delay between the proxy server and the remote server (100ms).

For the simulated HFC network, increasing the proxy server's initial TCP transmission window does not offer any additional gain in end-to-end performance. This behavior is attributable to the much longer round-trip time over the remote connection, which is comparable to the long initial waiting time on local connection. Consequently, even if the proxy server uses a larger initial window, packets do not arrive quickly enough over the remote connection for the local connection to exploit the early arrival of the second ACK from the subscriber's PC. The benefits of using a larger initial window at the proxy server become more apparent as the round-trip time over the remote connection decreases.

3.5 Performance in a Lossy Local HFC Network

In order to study the effectiveness of the proxy in dealing with packet loss in the local HFC access network, we next consider a scenario in which packet loss occurs in the HFC network during a 3MB data transfer from a remote server to a subscriber's PC. In this experiment, the remote connection is not subjected to any packet loss. As before, the TCP socket buffers for the local and the remote connection are both set to be 8KB.

The following observations are made from Figure 8 which contrasts the performance achieved when using a direct connection with that achieved when using a proxy:

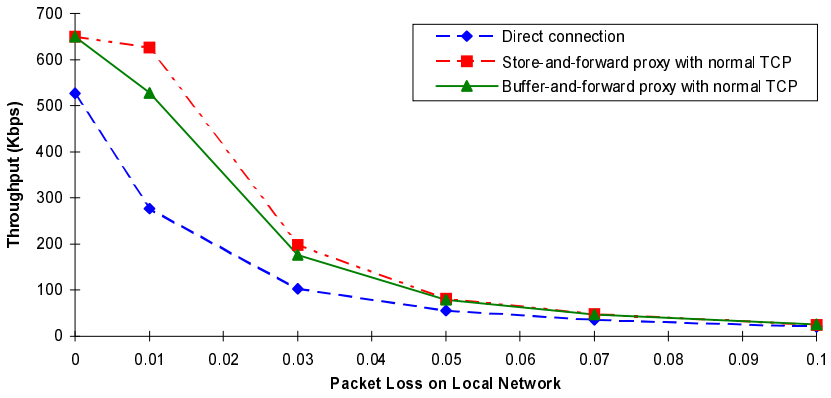


Figure 8. Illustration of the advantages of a proxy server in handling packet losses over the local HFC network. The data transfer size used was 3 MB.

- *Comparison of the performance of the proxy models with that of the direct connection:* When packet loss occurs, the proxy models degrade more gracefully in performance than the direct connection approach. The localization of packet loss detection and recovery enabled by the proxy servers is the main reason for their better performance under lossy HFC network conditions. When a direct connection is used, packets lost over the local HFC network segment are retransmitted all the way from the remote server. In contrast, both the store-and-forward proxy and the buffer-and-forward proxy handle packet losses that occur in the HFC network locally: Packets lost over the HFC network are detected and retransmitted by the TCP receiver at the proxy servers themselves, without involving the remote server. By doing so, the proxy servers not only avoid unnecessary retransmissions over the remote network segment, but they also enable faster recovery from packet loss. This is because the rate at which TCP's slow-start and fast retransmit algorithms increase the transmission window following a packet loss is directly related to the round-trip time of a connection. Since the proxy servers handle retransmissions in the local connection itself, the rate of the window increase following a packet loss is governed by the smaller round-trip time on the local connection.

  To illustrate the benefits of the faster window increase that the use of a proxy server enables, suppose that packet loss occurs over the local connection when the TCP transmission window at the proxy server is 8 packets. When fast retransmit detects the loss, the window shrinks to 4 packets. Then, about 4 round-trips, amounting to 80ms in time, are necessary for the local connection to return to at its peak rate. In contrast, if the same loss happened over a direct connection, it would cause the connection to operate below its peak rate for about 480ms (four round-trips again).

  A further advantage in using proxy servers is that at low packet loss rates, the reduction in throughput of the local connection caused by the packet losses may not be significant enough to force the local connection to be the bottle-neck that governs the end-to-end throughput. In such cases, the throughput achieved is not impacted by losses in the local HFC network. Figure 8 illustrates this feature for the store-and-forward proxy.

- *Comparison of the performance of the proxy models:* The differences between the proxy models becomes apparent when packet losses occur in the local network. During such times, frequent time-outs reduce the throughput of the local connection, forcing it to become the bottleneck that limits end-to-end throughput. In the case of the buffer-and-

forward proxy, when the local connection stalls for several seconds, the remote connection can proceed at its fastest rate only until the limited receiving socket buffer at the proxy becomes full. From this time, the remote connection has to remain idle until the local connection resumes. Following the time-out, the local connection resumes in the slow-start phase, with the transmission window being reset to one packet. Importantly, since the remote connection also remains idle for several seconds, the remote server's TCP stack may also drops its transmission window to one packet and re-initiates slow-start. While the local connection recovers quickly and begins operating at its maximum rate, because of its much larger round-trip time, the remote connection takes much longer than the local connection to begin operating at its maximum rate. In the interim period until the remote connection begins operating at its maximum rate, the local connection has to wait for packets to arrive over the remote connection before forwarding the packets to the subscriber's PC.

In contrast, the store-and-forward proxy guarantees complete independence between the local and remote connections. Since it provides intermediate storage for all packets received over the remote connection,the store-and-forward proxy ensures that even when the local connection is stalled, the remote connection can proceed at its fastest rate. Later, when the local connection recovers, it can proceed at its peak rate, forwarding packets from the proxy's intermediate storage until the end of the data transfer or until the local connection catches up with the remote connection. Because of the greater degree of independence it offers between the local and remote connections, the store-and-forward proxy server yields a higher throughput than a buffer-and-forward proxy server when packet losses occur in the local HFC network.

Figure 8 illustrates that the proxy servers are most effective at lower packet loss rates. At a 1% loss rate, the direct connection experiences almost a 50% reduction in throughput, whereas the store-and-forward proxy experiences less than a 5% drop in throughput. As packet loss increases above 3%, although the proxy models continue to perform better, their effectiveness degrades. The main reason for this change is that as the packet loss rate increases, the frequency of TCP time-outs in the local connection increase. Since the proxy servers only speed up the recovery after a time-out but not the duration of time-outs, as time-outs become more frequent, the faster recovery that the proxy servers enable becomes insignificant compared to the dramatic reduction in throughput caused by the large duration of the time-outs. Consequently, at higher loss rates, the proxy servers only yield marginally better performance than the direct connection.

3.6 Tuning TCP Implementations at Proxy Servers for Higher Performance

In the example in Figure 8, for the proxy models, the remote connection starts off by being the bottleneck that governs the end-to-end throughput obtainable. The degradation in end-to-end throughput with packet loss indicates that for the network configuration under study, the local connection becomes the bottleneck when packet losses occur in the local HFC network. To increase the throughput of the local connection during lossy network conditions, and thereby increase the end-to-end throughput, the two modifications proposed in [5] can be incorporated into TCP implementations at the proxy servers. To reduce the probability of occurrence of time-outs, the implementation of the fast retransmit strategy at the proxy server can be modified so that a TCP packet is retransmitted soon after the first duplicate ACK is received by the proxy server. A second modification is to reduce the duration of time-outs by using a 200ms retransmission timer instead of the current 500ms timer.

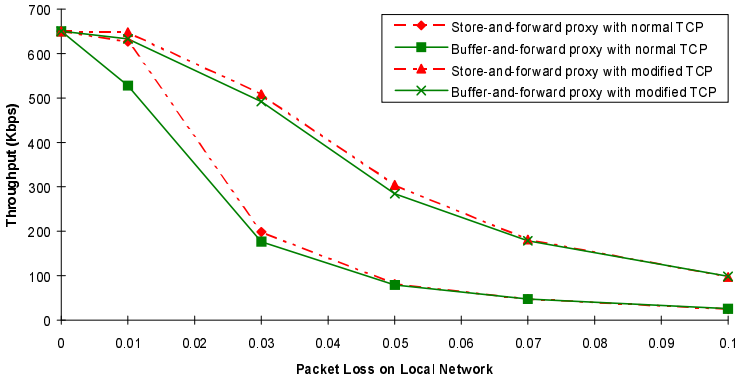


Figure 9. Illustration of the performance improvements that can be obtained by optimizing the TCP implementation at the proxy server for the lossy HFC access network. The data transfer size used in this experiment is 3 MB.

Figure 9 illustrates that these simple modifications to TCP implementations at proxy servers can yield significant performance improvements. For a 3 MB data transfer from a remote server, these modifications offer almost a two-fold increase in performance for both the store-and-forward and the buffer-and-forward proxy servers, when the loss rates are 3% and higher. Figure 10 also illustrates that with the above mentioned modifications to TCP implementations at the proxy, the buffer-and-forward proxy begins to perform almost as well as the store-and-forward proxy at all loss rates. The reason for this behavior is that the TCP modifications reduce the duration of time-outs experienced by the proxy for the local connection. Since the remote server uses the normal coarse granularity TCP retransmission timer, the local connection recovers from time-out much before the remote server drops its transmission window (determined based on the remote server's retransmission timer). Consequently, the remote connection can operate at its maximum window size soon after the local connection recovers from the time-out. As a result, the differences in performance between the two proxy models are reduced.

4 Factors Affecting the Performance of the Proxy Approaches

4.1 Variation in Performance with Location of Remote Servers

One of the factors governing the precise improvements obtained from the above TCP modifications is the round-trip time between the remote server and the local server complex (which houses the proxy server, if one exists). Assuming a 3% packet loss rate over the local HFC network, Figure 10 evaluates the performance benefits of using a store-and-forward proxy server to handle a 3MB data transfer from different remote servers. The following conclusions are drawn from this experiment:

- In general, an increase in round-trip time for a TCP connection causes a decrease in the rate of increase of the TCP transmission window. Even when the transmitter is operating at its maximum window, the longer round-trip time slows down the arrival of TCP ACKs at the transmitter, thereby lowering the throughput achieved. Figure 10 indi-

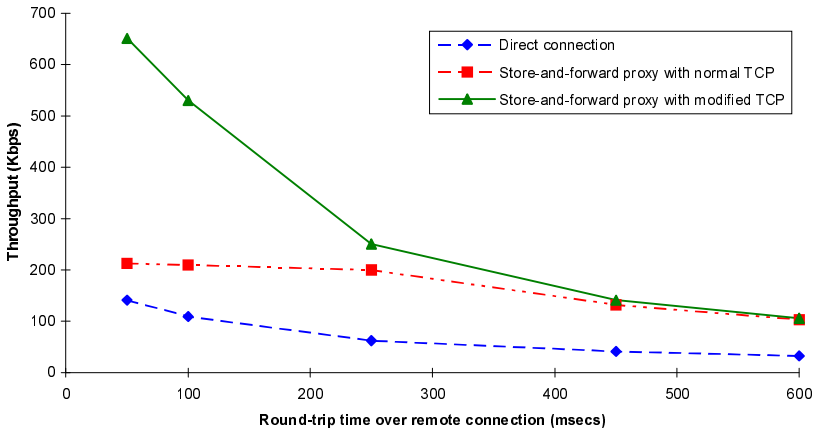


Figure 10. Comparison of the performance improvements obtained from using a proxy for access to remote servers with different RTTs. The data transfer size is 3MB and packet loss on the local HFC network is 3%.

cates that the throughput achieved over the direct connection drops with increase in round-trip time between the remote server and the local server complex.

- Figure 10 also illustrates that the proxy model consistently performs better than the direct connection. In fact, the end-to-end throughput for a proxy that implements normal (unmodified) TCP remains constant as the round-trip time over the remote connection increases to 300ms, after which point, the throughput drops with further increase in round-trip time. To explain this behavior, consider Figure 11, which compares the throughput achieved over the remote connection and the local connection in isolation, for different remote server locations. Since it is unaffected by the location of the remote server, the local connection yields a constant throughput. Unlike the local connection, the throughput achieved over the remote connection drops exponentially with increase in the round-trip time over the remote connection.

  Since the end-to-end throughput achieved when using a proxy server is dependent on whether the local connection or

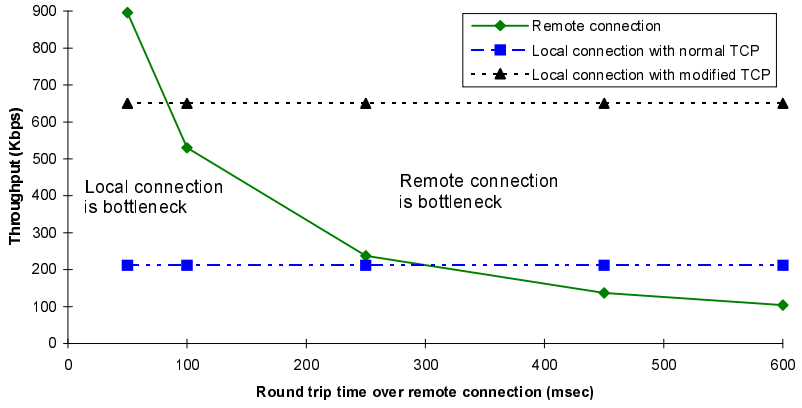


Figure 11. Comparisons of throughputs achieved over the remote connection and over the local connection in isolation, when using a store-and-forward proxy. The local network experiences a 3% packet loss, whereas the remote network is lossless

the remote connection is the bottleneck, from Figure 11, it is clear that when the round-trip time over the remote connection is less than 300ms and the proxy implements normal TCP, the local connection is the performance bottleneck. Consequently, in this same region in Figure 10, any change in the round-trip time for the remote connection has no effect on end-to-end throughput. Beyond this region, since the remote connection becomes the bottleneck, the end-to-end throughput drops with increase in round-trip time over the remote connection.

- As is evident from Figure 10, the advantages of tuning the proxy server's TCP implementation to handle losses drop with increase in round-trip time over the remote connection. Figure 11 indicates that when the proxy implements the modified TCP, the local connection is the bottleneck only until the round-trip time for the remote connection reaches 100ms. Beyond this region, since the remote connection becomes the performance bottleneck, the throughput achieved drops dramatically with increase in the remote connection's round-trip time. Correspondingly, the advantages from tuning TCP implementations at the proxy are reduced since the modifications to TCP only benefit the local connection. At the same time, however, the proxy continues to perform much better than the direct connection because it handles packet losses locally. In contrast, every time a packet loss occurs, the direct connection forces retransmissions all the way from the remote server, and the penalty of such retransmissions grows as the round-trip time over the remote connection increases.

4.2 Effect of Data Transfer Size

The performance improvements obtained also depend on the amount of data being transferred from the remote server to the subscriber's PC. Figure 12 compares the performance improvements obtained for different data transfer sizes under different HFC network conditions, for accesses to a remote server that has a round-trip time of 100ms to a store-and-forward proxy. At a low, 1% loss rate (Figure 12(a)), there is little difference in performance between a proxy that uses standard TCP and one that implements a TCP tuned for the lossy access network, for data transfers of 50KB and less. Even for 250KB transfers, the difference is less than 5%. The performance differential between the direct connection and the proxy is much higher. Almost all data transfers benefit from the proxy, with the performance improvement being 60% or more for data transfers of 100KB and more. As the loss rate increases, the performance of a proxy that implements standard TCP approaches that of the direct connection, whereas a proxy with a tuned TCP implementation proceeds to perform significantly better. At higher loss rates, even the smaller data transfers experience performance improvements. At a 5% loss rate (Figure 12(b)), a proxy with the modified TCP offers close to 100% improvement for data transfers of 100KB and above. At a 10% loss rate, a tuned proxy delivers nearly 200% improvement can befor data transfers as small as 20KB.

4.3 Buffer Sizing at the Proxy

Another optimization that can be performed at the proxy is to increase the TCP socket buffer size used at the proxy for the remote connection. Since the round-trip time over the remote connection is much larger than that for the local connection, by using a larger buffer size for the remote connection, the proxy permits the remote server to use a larger transmission window and thereby yields a higher throughput for the remote connection. If the remote connection is the performance bottleneck, the increase in throughput for the remote connection results in an increase in the end-to-end

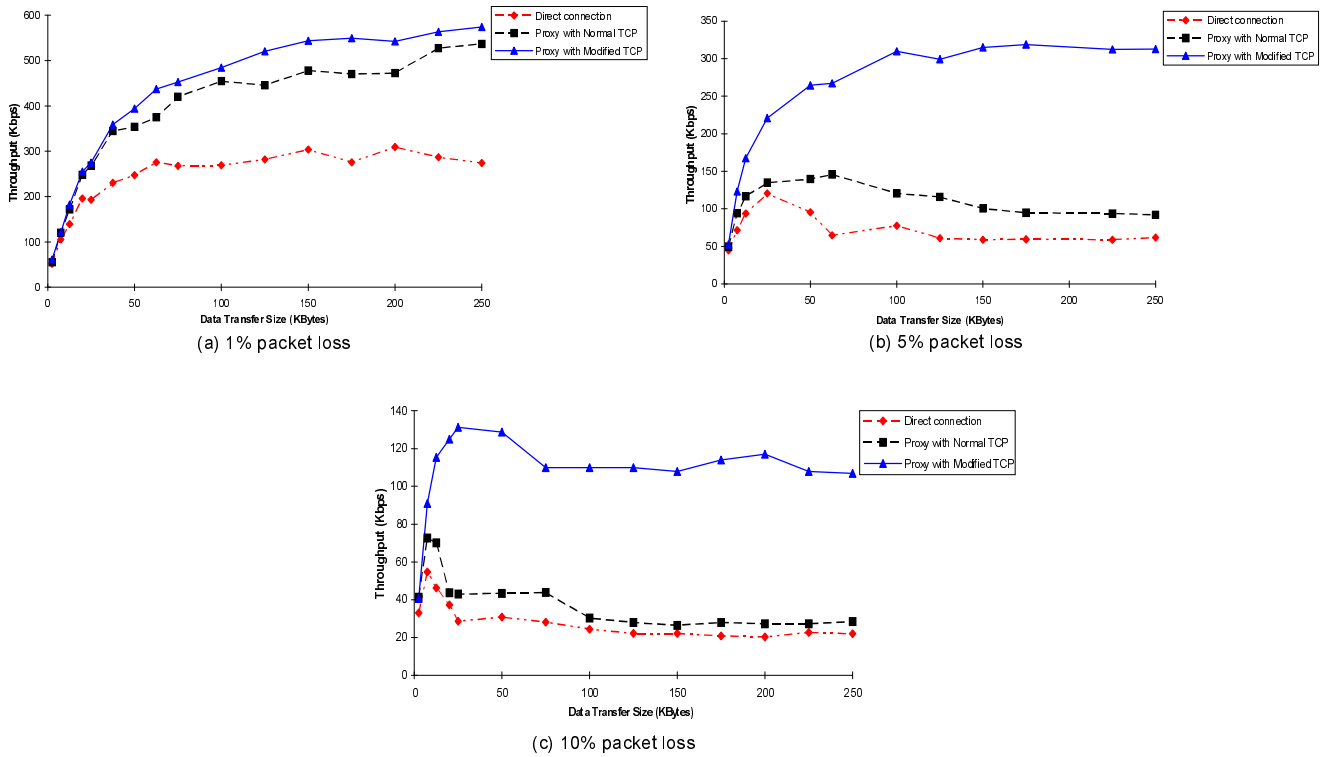


Figure 12. Comparison of the performance improvements obtained when using a store-and-forward proxy server for different data transfer sizes

throughput. An additional benefit when using the buffer-and-forward model proxy is that a larger socket buffer also speeds up recovery over the local connection following time-outs caused by packet losses in the local HFC network, since a larger number of packets is available at the proxy for transmission over the local connection.

Figure 13 illustrates that in the absence of packet loss, an almost four-fold increase in throughput can be achieved during a 3MB data transfer by increasing the socket buffer size for the remote connection from 8KB to 32KB for the simulated network configuration (Figure 8 depicts the performance when the remote connection's socket buffer size is 8KB). At low packet loss rates (< 2%), the larger, 32KB socket buffer is still effective with a proxy implementing normal TCP offering a significant increase in throughput compared to that obtained when using a direct connection. At higher loss rates, the frequency and duration of time-outs impact the achieved throughput to such an extent that the benefits of using a larger socket buffer at the proxy become insignificant. In this case, the modifications to the proxy's TCP implementation discussed in Section 3.6 offer more than three-fold improvement in performance. Figure 13 also illustrates that the larger socket buffer size reduces the dependence between packet transmissions over the remote and local connections, and as a result, further reduces the differences in performance between the buffer-and-forward proxy and the store-and-forward proxy approaches.

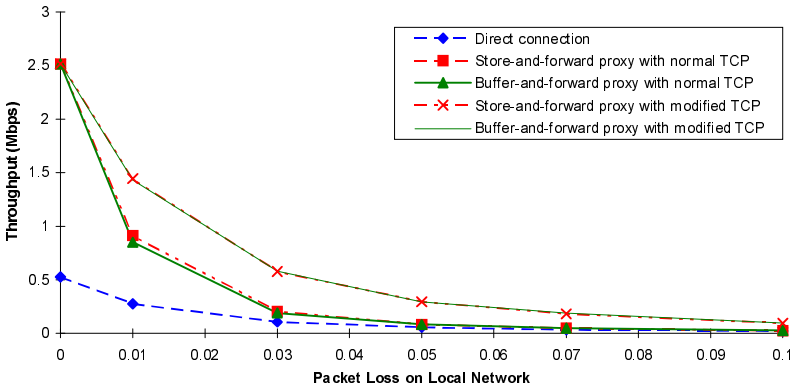
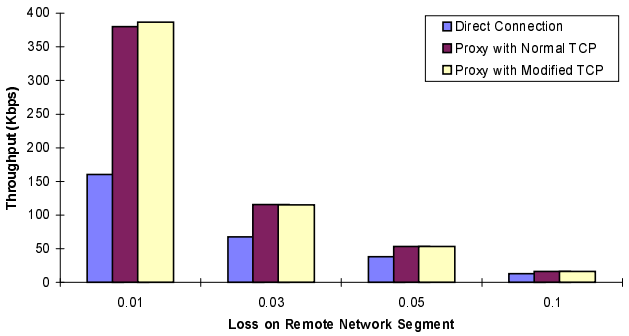


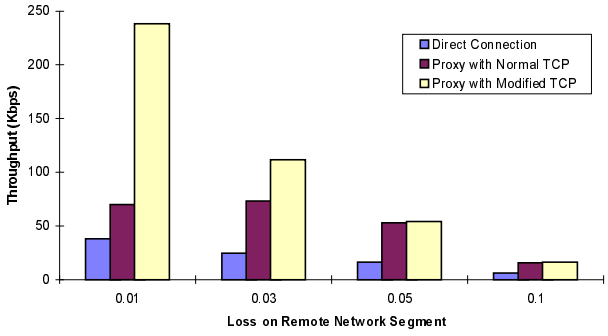
Figure 13. Performance improvements obtained by increasing the proxy server's socket buffer size for the remote connection from 8KB to 32KB

4.4 Performance When Losses Occur on the Internet

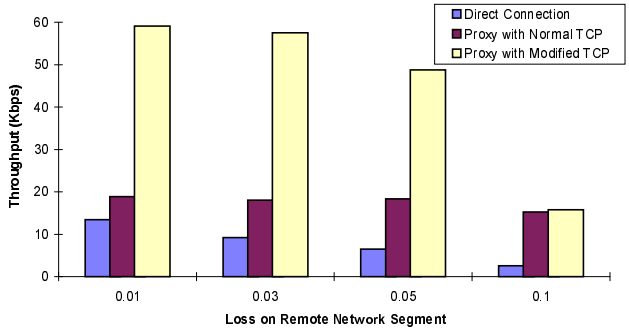
Having illustrated the effectiveness of a proxy in handling lossy access networks, we next evaluate the usefulness of the proxy in cases when packet losses occur not only in the local access network because of media errors but also in the



(a) 1% packet loss on local network segment



(b) 5% packet loss on local network segment



(c) 10% packet loss on local network segment

Figure 14. Comparison of the effectiveness of proxies for different packet loss rates in the local and remote network segments

Internet because of congestion. For a 3MB data transfer, Figure 14 compares the performance observed for different loss rates on the local and remote network segments. The following observations can be made from the figure:

- Since it has to retransmit packets all the way from the remote server, irrespective of whether the packet losses occur on the local or the remote network segments, the direct connection is equally affected by losses on the remote and local networks. The performance impact depends on the cumulative packet loss rate on the local and remote networks. For example, the throughput achieved by the direct connection when losses on the local and remote networks are 1% and 5%, respectively, is comparable to that achieved when the losses on the local and remote networks are 5% and 1%, respectively.
- Irrespective of the magnitude of the losses, the proxy approaches perform better than the direct connection. The advantages of the proxy in handling losses in the local network have already been highlighted in the previous sections. When losses occur in the remote network too, the proxy offers some advantages. By splitting the local and remote connections, the proxy permits losses in the local and remote networks to be handled independently. Moreover, since the round-trip time over the remote connection is smaller than the round-trip time over the end-to-end direct connection, the proxy enables faster recovery from losses that occur in the remote network. Since TCP time-outs occur if packet loss occurs when the transmission window is less than four packets, by providing larger buffering, the proxy permits the remote server to use a larger TCP transmission window, thereby reducing the probability of TCP time-outs over the remote connection.
- Figure 14 also illustrates that unlike the direct connection, the performance of the proxy approaches is dependent on the relative magnitudes of packet losses on the local and remote network segments. In general, the proxy approach performs better when the loss rate on the local network segment exceeds that on the remote network segement. For instance, when losses on the local and remote networks are 5% and 1%, respectively, the throughput achieved by a proxy that implements normal TCP is 70 Kbps. In comparison, when the losses on the local and remote networks are 1% and 5%, respectively, throughput drops to 50 Kbps. The reason for this difference is that the ability of the proxy to detect and recover locally from losses in the local network, without requiring retransmissions from the remote server, outweighs the advantages it offers for handling losses in the remote network. Furthermore, the enhancements to the proxy server's TCP implementation enable it to handle losses in the local network more efficiently than those in the remote network.
- The advantages of optimizing the proxy server's TCP implementation for HFC networks are apparent when the loss rate on the local network is greater than that on the remote network. Greater the difference between losses on the local and remote networks, greater the performance advantages from tuning the proxy server's TCP implementation to the HFC access network.

5 Conclusions

Our analysis has highlighted the performance advantages that can be obtained by using proxy servers to handle remote accesses in broadband data service deployments, even in cases when the proxies do not serve as data caches. Simula-

tions of typical HFC networks have indicated that by tailoring proxies for loss-prone hybrid fiber-coaxial access networks, several-fold increase in performance can be obtained.

As proxies begin to be used in broadband data service deployments, several critical issues in the design of proxies need to be further examined. The design of proxy architectures that scale to handle several thousand subscriber requests simultaneously and caching strategies that enhance performance of remote accesses even in loss-free network environments are areas for further research. Performance monitoring and fault diagnosis techniques for network environments in which a hierarchy of proxies is used to handle remote accesses is another topic for future investigation.

References

[1] M. Gray. Internet statistics: Growth and usage of the Web and the Internet. *Available from http://www.mit.edu/people/mkgray/net*, June 1996.

[2] NSFNET traffic distribution highlights. *Available from ftp://nic.merit.edu/statistics/nsfnet/1995/*, April 1995.

[3] J. Dail, M. Dajer, C.-C. Li, P. Magill, C. Siller, K. Sriram, and N. Whitaker. Adaptive digital access protocol: A MAC protocol for multiservice broadband access networks. *IEEE Communications Magazine*, 34(3):104–112, March 1996.

[4] C. Eldering, N. Himayat, and F. Gardner. CATV return path characterization for reliable communications. *IEEE Communications Magazine*, 33(8):62–69, August 1995.

[5] R. Cohen and S. Ramanathan. Tuning TCP for high performance in hybrid fiber coax networks. *Hewlett-Packard Laboratories Technical Report HPL-97-19*, September 1996.

[6] J. Limb and D. Sala. A protocol for efficient transfer of data over fiber/cable systems. In *Proceedings of IEEE INFOCOM'96, San Francisco, CA*, pages 904–911, March 1996.

[7] E. Perry and S. Ramanathan. Network management for residential broadband interactive data services. *IEEE Communications Magazine, Special Issue on Broadband Access*, November 1996.

[8] W. Stevens. *TCP/IP Illustrated, Vol. 1, Addison-Wesley, MA*, 1994.

[9] R. Braden. Requirements for Internet hosts – communication layers. *Internet Request for Comments, RFC 1122*, October 1989.

[10] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proceedings of 15th International Conference on Distributed Computing Systems (ICDCS)*, May 1995.

[11] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of ACM SIGCOMM'96, Palo Alto, CA*, pages 256–270, August 1996.

[12] S. McCanne and S. Floyd. ns, network simulator. *Available from http://www-nrg.ee.lbl.gov/ns/*, 1996.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. *Internet Engineering Task Force HTTP Working Group draft, draft-ietf-http-v11-spec-05*, June 1996.