Modeling TCP Performance with Proxies

Mingyan Liu, Navid Ehsan
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122, USA

Abstract— This paper investigates the TCP dynamics and performance over proxies that shorten the TCP feedback loop by segmenting the end-to-end connection. Such proxies are often used to improve TCP performance, e.g., a splitting/spoofing proxy in the satellite communication, and more commonly, a web cache. By analysis, we attempt to develop a basic understanding of the properties of TCP dynamics when such proxies are used, and further obtain certain design principles of systems involving such proxies. We present simple models capturing some features of the proxy performance in both the lossless and lossy scenarios. Due to the complexity involved, detailed analysis is only available in the lossless scenario, and our discussion in the lossy scenario is largely limited to steady state behavior. However we are able to obtain useful insight through such analysis. We identify conditions under which using a proxy provides significant or marginal performance gain by investigating factors including initial window size, congestion level of the proxy, and the level of asymmetry between the links segregated by the proxy. We also discuss how these conditions affect the deployment and provisioning of systems using proxies.

I. Introduction

THIS paper investigates certain type of proxies that cause changes in the TCP dynamics and the resulting performance implications. In particular, we focus on proxies that shorten the TCP feedback loop either by design or as a by-product. Such proxies are normally used to reduce the connection response time and achieve higher link utilization.

One typical example of such is a TCP connection splitting and spoofing proxy that pre-acknowledges the sender on behalf of the receiver (by spoofing the receiver's address), and forwards packets to the receiver on behalf of the sender (by spoofing the sender's address). Such a scheme is usually called split TCP, TCP spoofing [1] or indirect TCP (I-TCP) [2], [3]. It is commonly used in satellite communication to improve TCP performance over the large bandwidth-delay-product of satellite link since it can speed up the window growth and achieve higher capacity utilization, especially for short connections. It has also been proposed for terrestrial wireless networks (e.g., I-TCP) [3], [4], [5] as a means of separating the wired and the wireless part of the connection, and separating congestion losses and link failure losses. The motivation behind this approach is TCP's performance degradation in a heterogeneous environment. The idea is that if a communication path consists of physical medium that have very different characteristics, the end-to-end performance is optimized by isolating one type of physical link from another and optimizing each separately. However this approach generally violates the TCP end-to-end semantics, and will not work if the IP packets payload is encrypted [6].

Another example of such a proxy, which may seem less obvious, is a common web cache (e.g., with the web browser set to "proxy" mode). When there is a "hit" at the cache, the file is directly sent to the client from the cache. When there is a "miss", the cache opens up a connection to the remote server and starts downloading the file to the cache (for cacheable objects), while forwarding packets to the client at the same time. Thus the cache automatically "breaks" the server-client transfer into two separate connections [5]. In terms of TCP performance of the file transfer, this has exactly the same effect as split TCP (although the connection establishment is different). However, in this case the TCP semantics is preserved because the cache does not spoof the client's address, and so it acknowledges the server on behalf of itself rather than "pre-ack" on behalf of the client. Caching not only reduces latency by pushing the content closer to end users but also results in redirection of traffic that is meant for web servers, and can achieve better load balancing.

There has been implementation and experimental study of the TCP performance improvement using such proxies, especially split TCP in satellite and terrestrial wireless communications (e.g., [1], [2], [3]). In this paper we develop simple mathematical models to derive the TCP performance (mainly latency) when such a proxy is used, and analyze the level of performance improvement under different scenarios. Our motivation is three-fold: to have an analytical and quantitative study to gain insights into the dynamics of a shortened TCP loop in addition to simulation and experimental studies; to investigate the use of proxy as a general solution to problems involving heterogeneous links and large amounts of traffic; and more importantly, to apply such understanding to system level design issues.

In subsequent sections we will ignore whether the proxy spoofs addresses or not since it does not affect our analysis, and instead focus on a general model of server-proxy-client communication. Due to the complexity involved, detailed analysis is only available in the lossless scenario, and our discussion in the lossy scenario is largely limited to steady state behavior. However we are able to obtain useful insight through such analysis. In

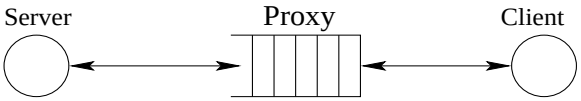

Fig. 1. Network Model

summary, we found that overall using the proxy results in higher utilization of the link capacity and lower latency. However, when the proxy becomes congested this performance gain is limited. In addition, when a connection is broken in two, the slower one always dominates the overall performance, and as this dominance increases, the gain from using the proxy is again reduced. These results imply that while optimization of separate parts of a connection (segregated by the proxy) is important, it is equally important to minimize the "asymmetry" between these parts, especially in a heterogeneous environment.

The organization of the paper is as follows. In Section 2 we present the network model and describe how the proxy functions. In Sections 3 through 5 we analyze the latency in file transfer with or without using the proxy. Two cases are investigated by assuming the links are lossless and lossy, respectively. The accuracy of our model is discussed. We then analyze the effect of initial window size, the congestion level of the proxy and the asymmetry between the two segments segregated by the proxy. Section 6 summarizes our results and concludes the paper.

II. Split Connection and the Network Model

A. Network Model

Our analysis is based on a two link model with one end host on each side and a proxy in the middle, as shown in Figure 1. In reality each of the two links may contain multiple intermediate routers and physical links, but are abstracted into a single link with a single round-trip time (RTT) parameter and a single loss rate parameter. In a real network, a spoofing proxy is usually placed between the wired part and the wireless (or satellite) link, and the client is usually located at the end of the wireless link. The location of a cache proxy is more arbitrary. File transfer is our main application of interest, and without loss of generality are considered to be from the server to the client.

When an end-to-end connection is established, the proxy functions as a normal router that forwards packets from the server to the client and vice versa. When connection splitting is used, the proxy acknowledges to the server, the client acknowledges to the proxy, and the proxy relays packets from the server to the client. Same procedure is used for the other direction of the connection. The two connections are inevitably coupled, but they keep separate sequence numbers and queues, and the proxy does not relay out-of-order packets from one to the other thus acting as a virtual source of the file. In general, with a spoofing proxy the initial connection establishment (three-way handshake) and the final closing are done in an end-to-end fashion. The connection is only split in two during the data transfer period, as shown in Figure 2. With a cache proxy there are two separate connection from the very beginning, i.e., three-way handshake is first conducted between the client and the proxy, and if there is a miss, another three-way handshake is conducted between the proxy and the server, as shown in Figure 3. Both situations result in approximately the same delay in connection establishment for a single connection. We therefore do no include this initial delay in our analysis and instead focus on the delay solely of data transfer, which is the duration between when the server sends the first data packet of a file and the time when the client receives the last data packet of a file. For a cache proxy if there is a hit on the file request, the content is retrieved directly from the cache. In this case the connection model is simply end-to-end from the client to the proxy, with a fraction of the entire server-client round-trip time. Our analysis therefore only applies to situations where there is a "miss".

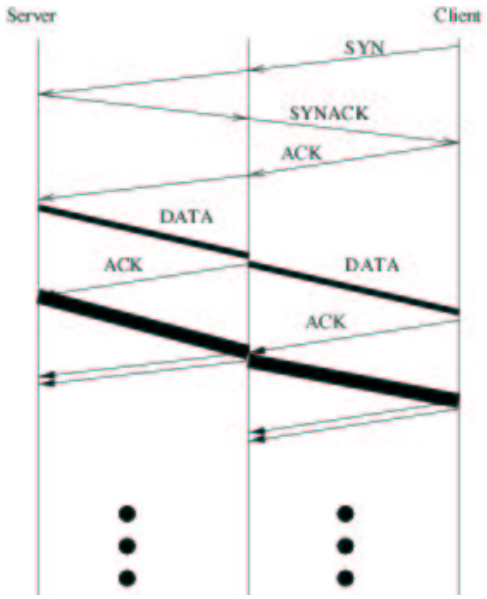

Fig. 2. File transfer using a splitting proxy

B. Assumptions and Parameters

We assume that a file contains exactly M segments of the maximum segment size (MSS). This is an approximation to an arbitrary file size whose last segment may be a fraction of MSS. However, this does not affect our method of analysis, and also does not affect the comparison between with or without using the proxy. We assume that both the end-to-end connection (server-client) and the split connections (server-proxy and proxy-client) have the same value of slow start threshold (ssthresh), W_{ss}, and the maximum window size W_{max}. These two values are also

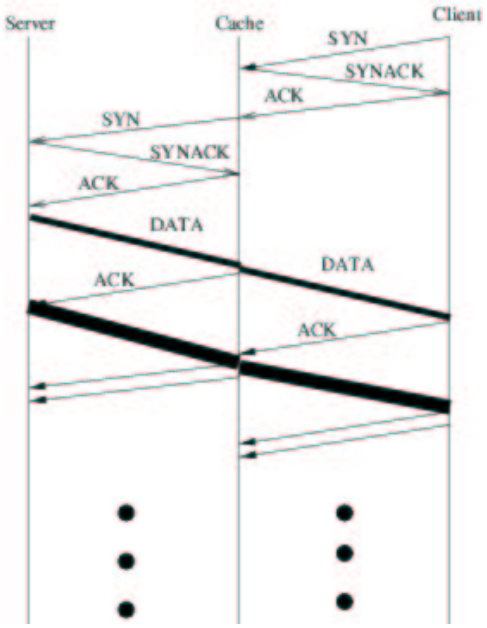

Fig. 3. File transfer using a cache upon miss

assumed to be in number of segments rather than number of bytes to simplify the analysis.

The server, the proxy and the client each has a transmission rate of C_1, C_p, and C_2, respectively. Assuming packet length of $D = MSS + 40$, (including both TCP and IP headers) the time it takes for the server to transmit a packet is $\mu_1 = \frac{D}{C_1}$, and $\mu_p = \frac{D}{C_p}$, $\mu_2 = \frac{D}{C_2}$ for the proxy and the client, respectively. When there are two separate connections, we assume a per-packet processing delay of t_p at the proxy. All other processing delays are ignored. We assume that each link has the same propagation delay and transmission rate in both directions. The one-way propagation delay on the server-proxy link and the proxy-client link are denoted by I_1 and I_2, respectively. Throughout our analysis, we assume that the transmission time of an ACK is negligible.

We further assume that the TCP sender is only constrained by the congestion window and not the advertised receive window size. Most work in TCP analysis assumes an infinite source, e.g., [7], [8], [9]. However, when we have two connections, the window of the second connection (proxy-client) evolves not only according to the window dynamics of TCP, but also according to the availability of packets (from the server-proxy connection), i.e., the first connection may not "catch up" with the second connection due to factors like initial window size, transmission rate, etc.. Therefore the window of the second connection will be forced to grow at a slower rate. We will discuss both in subsequent sections.

III. Lossless Links

Assuming that the window grows in the slow start and congestion avoidance stages until the maximum window size is achieved, the number of windows that is needed to cover a file of M segments can be calculated by extending the method presented in [10]. We also assume that delayed ACK is implemented. As shown in [11], since one ACK is generated for every b packets received before the timer expires, the rate of exponential growth of the congestion window is $r = 1 + \frac{1}{b}$, which equals 2 when no delayed ACK is used. Let w_o denote the initial window size. Let S be such that $w_o r^{S-1} < W_{sst} \leq w_o r^S$, if $M > \sum_{i=1}^{S} w_o r^{i-1}$, i.e., the slow start threshold *ssthresh* is reached during the $(S+1)^{th}$ window if the file is big enough. Similarly, let M_x be such that $W_{sst} + \frac{M_x - S - 1}{b} < W_{max} \leq W_{sst} + \frac{M_x - S}{b}$, i.e., the maximum window size is achieved during the $(M_x+1)^{th}$ window if the file is big enough. All subsequent windows have the same window size of W_{max}. The number of windows needed to transfer a file is then given by the following:

$$K = \begin{cases} \min\{k : \sum_{i=1}^{k} w_o r^{i-1} \geq M\} \text{ if } k \leq S \\ \min\{k : \sum_{i=1}^{S} w_o r^{i-1} + \sum_{i=S+1}^{k} (W_{sst} + \frac{i-S-1}{b}) \\ \qquad \geq M\} \text{ if } S < k \leq M_x \\ \min\{k : \sum_{i=1}^{S} w_o r^{i-1} + \sum_{i=S+1}^{M_x} (W_{sst} + \frac{i-S-1}{b}) \\ \qquad + \sum_{i=M_x+1}^{k} W_{max} \geq M\} \text{ if } M_x < k \end{cases} .$$

A. Delay Models

We first consider an end-to-end connection between the server and the client. Assuming that the links are lossless and that connections are only constrained by congestion window size, after the server sends a window's packets it waits for the first ACK to come back, if it takes longer for the ACK to arrive than it takes to transmit the window's worth of data. The time it takes to transmit the k^{th} window is a function of the packet transmission time at the sender given by

$$t_k(\mu_1) = \begin{cases} w_o r^{k-1} \mu_1 & \text{if } k \leq S \\ (W_{sst} + \frac{k-S-1}{b}) \mu_1 & \text{if } S < k \leq M_x \\ W_{max} \mu_1 & \text{if } M_x < k \end{cases} . \quad (1)$$

Therefore if $\mu_1 \geq \mu_p$, which indicates that the proxy transmits at least as fast as the server and thus packets will not experience queueing delay at the proxy, the round-trip time of the end-to-end connection is $2(I_1 + I_2)$. We define R_e to be the time it takes for the first ACK to arrive after the first packet was sent, thus $R_e = \mu_1 + \mu_p + 2(I_1 + I_2) + (b-1)\mu_1 = b\mu_1 + \mu_p + 2(I_1 + I_2)$. Note that this expression assumes that there are at least b packets in a window so that the receiver can immediately return an ACK upon receipt of the b^{th} packet. If for example $w_o = 1$ and $b = 2$, then the receiver may have to wait for the delayed ACK timer to expired to return an ACK. In the rest of our analysis we will ignore this difference,

which can be easily taken into account. The total time it takes to transfer the file is then

$$T_e(M) = M\mu_1 + \sum_{k=1}^{K-1}[R_e - t_k(\mu_1)]^+ + I_1 + I_2 + \mu_p, \quad (2)$$

where $[a]^+ = a$ for a positive and 0 otherwise. This latency reflects the total transmission time, the time that the server spends waiting for ACKs, and the time for the last window to reached the client.

When $\mu_1 < \mu_p$, packets could build up at the proxy waiting to be transmitted into the slower link and experience additional queueing delay at the proxy. In this case the ACKs of the same window arrive at the server approximately μ_p apart instead of μ_1, thus the server may need to wait for every ACK of the same window instead of stalling after sending out the entire window. We derive the latency by examining from the client's side. Since $\mu_p > \mu_1$, the client receives packets of the same window continuously at rate $1/\mu_p$. The time that the client is idle is therefore $[\mu_1 + \mu_p + 2(I_1 + I_2) + (b-1)\mu_p - t_k(\mu_p)]^+$, where $t_k(\mu_p)$ is the time it takes the client to receive the k^{th} window, and $t_k(\cdot)$ is given by (1). The latency is then

$$T_e(M) = M\mu_p + I_1 + I_2 + \mu_1 + \sum_{k=1}^{K-1}[\mu_1 + 2(I_1 + I_2) + b\mu_p - t_k(\mu_p)]^+ \quad (3)$$

which reflects the time the client spends receiving the file, waiting for the next window, and the time for the first window to reach the client. Redefining $R_e = \min(\mu_1, \mu_p) + b\max(\mu_1, \mu_p) + 2(I_1 + I_2)$, (2) and (3) can be combined into

$$T_e(M) = M\max(\mu_1, \mu_p) + I_1 + I_2 + \min(\mu_p, \mu_1) + \sum_{k=1}^{K-1}[R_e - t_k(\max(\mu_1, \mu_p))]^+.$$

When the proxy is used, we have two serial connections. Note that these two connections are not independent but coupled by data. This is because the second connection (proxy-client) cannot send any data packets it has not received from the first connection (server-proxy) and therefore be constrained. This can be caused by a much larger initial window size and/or a much shorter round-trip time on the second connection. In this scenario the second connection has a limited source based on the sending of the first connection. In [12] we developed a detailed model for this scenario. Due to space limit we do not present it here. However, similar qualitative insight can be obtained without having to go through the detailed analysis. For the rest of our discussion we will assume that the second connection is never constrained by the first connection, which could imply $w_o \geq w_o^{'}, \mu_1 \leq \mu_p$ and/or $I_1 \leq I_2$, where $w_o^{'}$ is the proxy's initial window size.

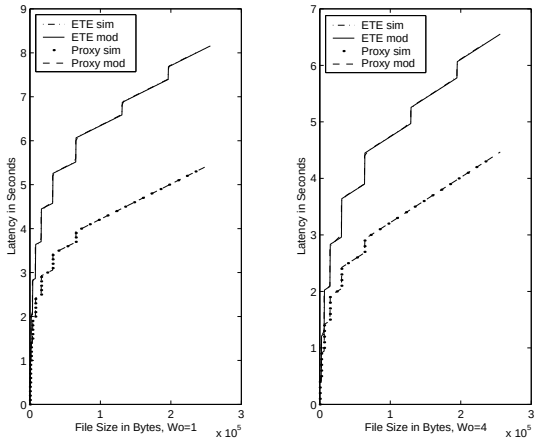

Fig. 4. Latency vs. file sizes, with initial window size of 1 and 4, respectively.

The proxy receives the first packet from the server at time $\mu_1 + I_1$. Assuming there is t_p delay for processing at the proxy, the proxy starts sending this packet to the client at time $\mu_1 + I_1 + t_p$. From this point on, we only need to focus on the second connection since the latency is only determined by this connection. By following the same analysis, we have the total latency for the proxy case

$$T_p(M) = \mu_1 + I_1 + t_p + M\mu_p + \sum_{k=1}^{K^{'}-1}[R_2 - t_k(\mu_p)]^+ + I_2, \quad (4)$$

where $t_k(\cdot)$ is given in (1), $K^{'}$ is the total number of windows needed for the transfer, and $R_2 = \mu_p + 2I_2 + (b-1)\mu_p = b\mu_p + 2I_2$ is the time it takes for the ACK to come back to the proxy. This latency reflects the initial delay for the first packet to arrive at the proxy, the total transmission time at the proxy, stall time and the time for the last packet to reach the client.

B. Validation

Figure 4 compares the numerical results from our model with *NS2* simulation, for both the end-to-end and proxy schemes. In this case, $C_1 = C_p = C_2 =$ 1Mbps. The initial window size is set to 1 and 4, respectively. Unless pointed out explicitly, our numerical results and simulation throughout this paper are based on the following parameters: MSS=512 bytes, ssthresh=128 segments. In both cases, $I_1 = 150$ ms, $I_2 = 250$ ms. Each graph contains four curves, two from ns simulation (sim) and two from our model (mod). We see that each pair (sim and mod) overlaps almost completely.

IV. Links with Random Losses

When losses (either due to congestion or link failure) are present the analysis becomes more complicated. Moreover the analysis is largely limited to the steady state

study of TCP connections which is applicable in the case of an unlimited file transfer, less accurate in the case of a finite TCP connection, and much less in the case of a proxy as shown in this section.

A. The Server-Proxy Link is lossless

If we assume that the server-proxy is lossless, then the methods introduced in [9], [8], [7] can be applied to determine the throughput and delay of the proxy-client connection. In particular, the TCP bulk data transfer throughput is shown to be well approximated by $\lambda(RTT, p) = \sqrt{\frac{3}{2bp}}\frac{1}{RTT}$, where p is the probability of loss for a single packet at low loss rate [9], and is more accurately approximated in [8] by considering timeouts. These results were developed for bulk TCP transfers and were based only on analysis of the TCP congestion avoidance phase. In [7] it was shown that they can be equally effective when applied to short TCP connections if combined with slow start analysis.

The performance implication of using a proxy when losses are present immediately follows: these results show that the throughput of a TCP transfer is inversely proportional to the connection round-trip time and the square root of the loss rate. If losses are concentrated on the proxy-client link, then using the proxy effectively isolates the part of the connection that involves loss, and reduces the round-trip time required to recover the losses, thus achieve higher throughput and lower latency. The same key concept can be seen in schemes such as Snoop TCP [13], [14], WTCP [15], [16], and [17], [18] that use local retransmission (some at the link layer, some at the transport layer).

Specifically, denoting the loss rate on the proxy-client link by p_2, and the throughput by $\lambda(RTT, p_2)$, the transfer latency of a file of size M using end-to-end connection is given by

$$\begin{aligned} T_e &= \sum_{n=0}^{M} p(n)(T_e(n) + \frac{M-n}{\lambda(RTT, p_2)}) \\ &= \sum_{n=1}^{M} p(n)T_e(n) + \frac{M - m_{loss}}{\lambda(2(I_1+I_2), p_2)} \end{aligned} \quad (5)$$

where $p(n) = (1-p)^n p$ for $n < M$; $p(n) = (1-p)^M$ for $n = M$ denotes the probability that n packets are successfully sent before the first loss occurs. $m_{loss} = \frac{(1-(1-p_2)^M)(1-p_2)}{p_2} + 1$ is the expected number of packets sent before the first loss occurs. $T_e(\cdot)$ is the latency function of an end-to-end connection shown in Section 3.

When using the proxy, assuming that the proxy-client connection is not constrained by the server-proxy connection (e.g., $I_1 < I_2, \mu_1 < \mu_2$),

$$T_p = \sum_{n=1}^{M} p(n)T_p(n) + \frac{M - m_{loss}}{\lambda(2I_2, p_2)}. \quad (6)$$

The difference of the two (using results from Section 3) $T_e - T_p$ is

$$\begin{aligned} &\sum_{n=1}^{M} p(n)(T_e(n) - T_p(n)) + \frac{(M - m_{loss})\sqrt{2bp_2}}{\sqrt{3}}(2I_1) \\ &\approx (\sum_{n=1}^{M} p(n)(k_n - 1) + \frac{(M - m_{loss})\sqrt{2bp_2}}{\sqrt{3}})(2I_1) \end{aligned}$$

where k_n is the number of window needed to cover a file of n segments. The approximation in the last equation is based on the assumption that the link capacity is not filled during the transfer of n packets. For a given file size and loss rate on the proxy-client link, the first term of the above equation is a constant, and the amount of gain in using the proxy depends on the round-trip time of the server-proxy connection.

Figure 5 compares the latency obtained using this analysis with the result from simulation. Simulation is the average over running 50 independent simulations. There is an obvious discrepancy between the two curves. This is mainly due to the fact that the delay model basically assumes that the connection goes into steady state right after the first loss. The two curves eventually approach each other as the file size increases (number of packets sent in this figure). This is because for a large file transfer the effect of the above assumption is diluted (since the effect of steady state will dominate).

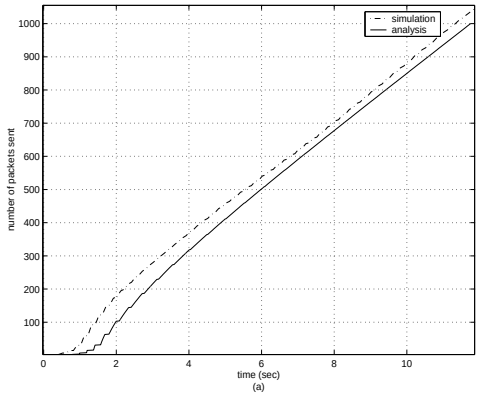

Fig. 5. Latency when splitting is used, where the first link lossless, $R_1 = 100msec, R_2 = 200msec, p_2 = 0.005, W_{sst} = 40$

B. Random Losses on Both Links

When losses are present on both links, using the previous analysis for both connections provides only gross approximation. Suppose the loss rates on the server-proxy link and the proxy-client link are p_1 and p_2, respectively. Assuming losses are independent, the overall loss rate experienced by an end-to-end connection is $p = p_1 + p_2 - p_1 p_2$. For an infinite file transfer, in the long

run the server-proxy connection has an average throughput $\lambda_1 = \sqrt{\frac{3}{2bp_1}}\frac{1}{R_1}$ and the proxy-client connection has an average throughput $\lambda_2 = \sqrt{\frac{3}{2bp_2}}\frac{1}{R_2}$ if unconstrainted by the server-proxy connection. The slower one of the two is going to dominate the combined throughput and delay. However, both values are greater than the throughput of the end-to-end connection $\lambda = \sqrt{\frac{3}{2bp}}\frac{1}{R_1+R_2}$ since it has a larger loss rate and a larger round-trip time. Therefore by segregating the server-client connection into parts that each has a smaller loss rate and round-trip time, using the proxy achieves higher throughput and thus lower latency.

For a finite file transfer, the latency of the end-to-end connection is given by (5) with loss rate p. When using the proxy, we consider m_1, m_2, the expected number of packets sent successfully before the first packet loss occurs to the two connections, respectively. The two connections have bulk transfer throughput $\lambda_1 = \lambda(2(I_1 + I_2) + \mu_p, p_1), \lambda_2 = \lambda(2I_2, p_2)$, respectively. Following this we can characterize the sending process and congestion window evolution of the server for the first m_1 packets, and λ_1 as an approximation to the remaining of the transfer. The characterization of the proxy can be obtained by listing all possible cases comparing m_1 and m_2, $T_e(m_1)$ and $T_p(m_1)$, λ_1 and λ_2. Figure 6 compares the results obtained from this analysis to the simulation results. It can be seen that the two do not match for a wide range of file sizes. Indeed the discrepancy seems to diverge. Our conclusion is the current model of TCP throughput and latency is insufficient to accurately predict the proxy performance with losses on both segments. The development of a more suitable model is part of our on-going research.

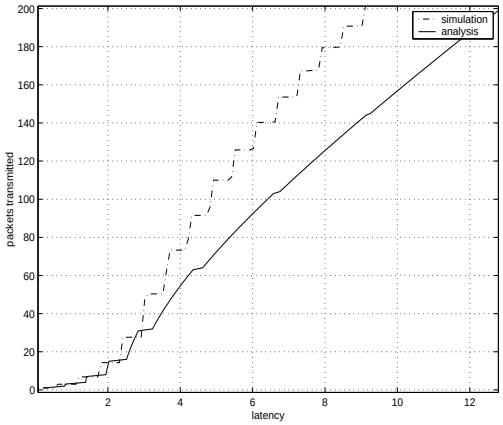

Fig. 6. Latency when splitting is used with losses on both links, $R_1 = 200msec, R_2 = 400msec, p_1 = 0.01, p_2 = 0.005$

V. Analysis and Discussions

In this section we focus on the model developed for the lossless scenario. This is not a realistic scenario, nevertheless insights are obtained via certain simplification and approximation especially for short file transfers.

TABLE I

Initial window size of the end-to-end connection

File Size (KBytes)	10	20	25	30	35
w_o	5	7	8	9	9

A. Initial Window Size

From the definition of S and M_x, $S = \lceil \log_r(\frac{W_{ss}}{w_o}) \rceil$, $M_x = b(W_{max} - W_{ss}) + S$. Consider a file that finishes transferring within the slow start phase, the total number of windows needed to cover the file, K, would be such that $K \leq S$. Since $M \leq \sum_{i=1}^{K} w_o r^{i-1} = w_o \frac{r^K - 1}{r-1}$, this means $M \leq \frac{W_{ss} - w_o}{r-1}$. K is the smallest integer that satisfies $M \leq w_o \frac{r^K - 1}{r-1}$, therefore, $K = \lceil \log_r(\frac{M}{w_o}(r-1)+1) \rceil \approx \log_r(\frac{M}{w_o}(r-1)+1)+1$. Assume $\mu_1 W_{ss} \leq R_e$, i.e., the link (or pipe capacity) is not filled during slow start, and $\mu_1 < \mu_p$ that the proxy is slower than the server, we have $T_e =$

$$M\mu_p + (K-1)R_e - \sum_{k=1}^{K-1} w_o r^{k-1}\mu_p + I_1 + I_2 + \mu_1$$
$$\approx R_e \log_r(\frac{M}{w_o}(r-1)+1) + C_1, \quad (7)$$

where $C_1 = I_1 + I_2 + \mu_1$. This last equation is the same as presented in [7], but derived in a different way.

Similarly, for an initial window size $w_o^{'}$ used by the proxy, assuming $\mu_p W_{ss} \leq R_2$, we have $K^{'} = \lceil \log_r(\frac{M}{w_o^{'}}(r-1)+1) \rceil \approx \log_r(\frac{M}{w_o^{'}}(r-1)+1)+1$ and

$$T_p \approx R_2 \log_r(\frac{M}{w_o^{'}}(r-1)+1) + C_2, \quad (8)$$

where $C_2 = I_1 + I_2 + \mu_1 + t_p$. Note that $C_1 \approx C_2$ and both are close to one half of R_e. Since $R_e > R_2$, w_o has to be great than $w_o^{'}$ in order to achieve the same delay. More specifically, $(\frac{M}{w_o}(r-1)+1)^{R_e} \approx (\frac{M}{w_o^{'}}(r-1)+1)^{R_2}$ leads to $w_o = \frac{M(r-1)}{(\frac{M}{w_o^{'}}(r-1)+1)^{R_2/R_e}-1}$. Table I shows some values of M and w_o based on this approximation for $r = 2, w_o^{'} = 1, R_2/R_e = 0.5$ and $W_{ss} = 128$ segments. We see that in order to achieve similar latency even for relatively small files we need significantly larger initial window size for the end-to-end connection.

B. Slow or Congested Proxy

When the same initial window size w_o is used, the difference in delay between the two is

$$T_e - T_p = (R_e - R_2)\log_r(\frac{M}{w_o}(r-1)-1)$$
$$= (2I_1 + \mu_1)\log_r(\frac{M}{w_o}(r-1)-1) \quad (9)$$

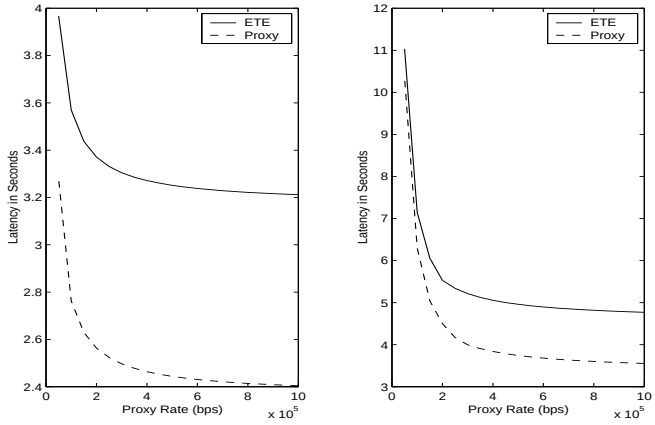

Fig. 7. Latency vs. C_p, the transmission rate of the proxy. File size is 11 Kbytes and 51 Kbytes, for the graph on the left and right, respectively. $C_1 = C_2 = 1$ Mbps in both cases. These graphs are derived from our model.

This difference increase as M and I_1 increase, but seems invariant to changes in μ_p. As μ_p increases, which corresponds to a slower proxy, the difference in delay remains constant so long as $\mu_p w_o r^{k-1} \leq R_2$ for any $k \leq K$. However, as μ_p keeps increasing to the point where the pipe is filled before the file transfer completes, the difference quickly reduces. In particular, if the pipe capacity is achieved during the k_e^{th} window for the end-to-end connection, i.e., $\mu_p w_o r^{k_e - 1} \geq R_e$, then $k_e = \lceil \log_r(\frac{R_e}{\mu_p w_o}) + 1 \rceil$ and

$$\begin{aligned} T_e &\approx M\mu_p + R_e \log_r(\frac{R_e}{\mu_p w_o}) - \frac{R_e - \mu_p w_o}{r-1} \\ &\quad + I_1 + I_2 + \mu_1. \end{aligned} \tag{10}$$

We can get a similar expression for T_p, and thus

$$\begin{aligned} T_e - T_p &\approx R_e \log_r(\frac{R_e}{\mu_p w_o}) - R_2 \log_r(\frac{R_2}{\mu_p w_o^{'}}) \\ &\quad + \frac{\mu_p(w_o - w_o^{'}) - 2I_1 - \mu_1}{r-1}. \end{aligned} \tag{11}$$

This expression decreases as μ_p increases. This result can be clearly observed in Figure 7. A slower proxy (increased μ_p and decreased transmission rate) can be viewed as an approximation to a busier or more congested proxy, because under such situation each TCP connection only gets a fraction of the total proxy capacity (assuming the proxy has sufficient buffer), and queuing is increased. This result shows that as the proxy becomes busy, the gain from using separate connections reduces because the bottleneck dominates the overall performance no matter which scheme we use. In a system where a proxy is placed at the aggregation point of incoming traffic, adequate provisioning of such a proxy becomes very important since otherwise very little is gained from using a proxy.

C. File Size

In the case where the file transfer enters the congestion avoidance stage, i.e., $M > \frac{W_{ss} - w_o}{r-1}$, similar analysis apply and we have $T_e = M\mu_p + R_e(k_e - 1) - \sum_{k=1}^{k_e - 1} t_k(\mu_p) + I_1 + I_2 + \mu_1$ and $T_p = M\mu_p + R_2(k_e^{'} - 1) - \sum_{k=1}^{k_e - 1} t_k(\mu_p) + I_1 + I_2 + \mu_1 + t_p$, where k_e and $k_e^{'}$ are the total number of window sent before the pipe becomes full for the end-to-end and the split connection, respectively. In case when the file finishes transfer before the pipe is full, $k_e = k_e^{'} = K$, and the difference between the two is mainly $(R_e - R_2)(K-1)$, which increases as M increases (K increases with M). However, if M is large enough and the pipe is filled up before the transfer completes, then the difference between the two stays constant, and both increase with rate μ_p as the file size increases. This can be observed in Figure 8.

D. Connection With Asymmetric Segments

Suppose $w_o \leq w_o^{'}$ and $R_1 > R_2$, i.e., the proxy-client connection is constrained by the server-proxy connection throughout the entire file transfer. Further assume that the file transfer is only limited to slow start phase. Following our previous analysis, we can show

$$\begin{aligned} T_p(M) &= R_1 \log_r(\frac{M(r-1)}{w_o} + 1) - I_1 + I_2 \\ &\approx R_1 \log_r(\frac{M(r-1)}{w_o} + 1), \end{aligned}$$

with an error within half of R_1.

Suppose we now let $R_1 < R_2$, but keep $R_1 + R_2$ unchanged, and let $w_o > w_o^{'}$, then using our earlier analysis we get $T_p(M) = R_2 \log_r(\frac{M(r-1)}{w_o^{'}} + 1)$ while the latency of end-to-end connection remain the same $T_e(M) = (R_1 + R_2)\log_r(\frac{M(r-1)}{w_o} + 1)$. We see that when using the proxy, the longer connection of the two ($\max\{R_1, R_2\}$) determines the total latency. As the difference between the two round-trip times increases, the gain from using the proxy reduces. Under this scenario the performance of the proxy is maximized when the two connections are "similar" – $R_1 \approx R_2, w_o \approx w_o^{'}$. Same argument can also be derived for file transfers that enter congestion avoidance phase. This is an interesting observation considering the fact that many such proxies are used or proposed to be used in a heterogeneous environment where links have very different properties. This indicates that while it is very important to optimize each link separately, it is equally important to minimize the asymmetry between links since if separate optimization only increases the difference, e.g., making the fast link even faster, the resulting performance might not be improved.

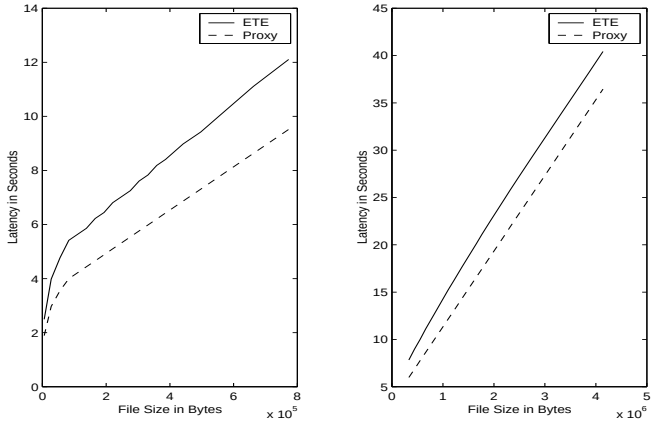

Fig. 8. Latency vs. file size. For small files the latency of an end-to-end connection increases faster than that of split connections. However as file size grow big enough to fill up the capacity, the two have same growth rate and the difference stays constant. These graphs are derived from our model.

VI. Discussions and Conclusions

In this study we examined using proxy as a way of improving TCP performance in various situations. Such proxies typically break an end-to-end connection into two segments, such as a spoofing proxy. We developed models to investigate the TCP dynamics when proxies are used and compared its performance with end-to-end TCP connection. We summarize our observations and conclusions as follows.

In general using proxy (or separate TCP connections) results in lower latency from our analysis. For an end-to-end connection this can be compensated by increasing the initial window size. However, as we show in Table I it requires significantly larger initial window size even for reasonably small file sizes, which makes it less practical in real applications. When the proxy becomes the bottleneck, the gain from using the proxy quickly diminishes. In systems where such a proxy is positioned at a place that all connections have to go through, e.g., in a satellite system the proxy is co-located with the satellite gateway so that all connections go through the proxy, the performance gain from the proxy can be limited especially during busy hours. This may also cause buffer overflow at the proxy, or cause the proxy to advertise smaller receive window size, which we did not consider explicitly in this paper. When this is the case, queueing becomes severe and packets quickly build up at the proxy, especially if in addition I_1 is much smaller than I_2. It is therefore important to properly provision such systems and implement some form of dynamic flow control at the proxy. This may be prevented by using a much larger initial window size over the proxy-client link.

A proxy achieves the effect of localizing error/loss recovery and in general improves the throughput and reduces latency of a connection when losses are present. One of the common situations where proxies are used is a heterogeneous environment where parts segregated by the proxy have very different link characteristics, e.g., propagation delay, loss rate, etc.. Interestingly, the performance gain in using a proxy is maximized when both parts have similar properties. Since the slower part always dominates the overall performance, as the level of asymmetry increases, the performance gap between using a proxy and using an end-to-end connection becomes smaller. This implies that while it is important to separately optimize these heterogeneous parts of the connection, it is also important that such optimization reduces the asymmetry between them.

References

[1] V. G. Bharadwaj, "Improving TCP Performance over High-Bandwidth Geostationary Satellite Links," Tech. Rep. MS 99-12, Institute for Systems Research, University of Maryland, College Park, 1999, http://http://www.isr.umd.edu/TechReports/ISR/1999/.

[2] A. Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. IEEE ICDCS*, pp. 136–143, 1995.

[3] A. V. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Transactions on Computers*, vol. 46, no. 3, pp. 260–278, 1997.

[4] K. Brown and S. Singh, "A Network Architecture for Mobile Computing," *IEEE INFOCOM*, pp. 1388–1396, 1996.

[5] S. Sibal P. Rodriguez and O. Spatscheck, "TPOT: Translucent Proxying of TCP," Tech. Rep., AT & T labs-Research and EURECOM Technical Report, 2000.

[6] M. Karir, "IPSEC and the Internet," Tech. Rep. MS 99-14, Institute for Systems Research, University of Maryland, College Park, 1999, http://http://www.isr.umd.edu/TechReports/ISR/1999/.

[7] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," *IEEE INFOCOM*, 2000.

[8] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE Trans. Networking*, vol. 8, no. 2, pp. 133–145, 2000.

[9] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE Trans. Networking*, vol. 5, no. 3, pp. 336–350, 1997.

[10] J. Kurose and K. Rose, *Computer Networking, A Top-Down Approach Featuring the Internet.*

[11] M. Allman and V. Paxson, "On Estimating End-to-end Network Path Properties," *SIGCOMM*, 1999.

[12] M. Liu and N. Ehsan, "Modeling TCP performance with proxies," *Technical Report, EECS Department, University of Michigan, Ann Arbor*, 2001.

[13] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP Performance Over Wireless Networks," *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'95)*, vol. 2, no. 11, 1995.

[14] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Trans. on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[15] K. Ratnam and Ibrahim Matta, "WTCP: An Efficient Mechanism for Improving TCP Performance Over Wireless Links," *Proc. IEEE ISCC*, pp. 74–78, 1998.

[16] K. Ratnam and Ibrahim Matta, "Effect of Local Retransmission at Wireless Access Points on The Round Trip Time Estimation of TCP," *Proc. 31st Annual Simulation Symp.*, pp. 150–156, 1998.

[17] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP Performance Over Wireless Network at The Link Layer," *ACM Mobile Networks & Applications Journal*, 1999.

[18] C. Parsa and J. J. Garcia-Luna-Aceves, "TULIP: A Link-Level Protocol for Improving TCP over Wireless Links," *Proc. IEEE WCNC'99*, pp. 1253–1257, 1999.