Internet Service via Broadband Satellite Networks

Norman P. Butts, Lockheed Martin Global Telecommunications
Vijay G. Bharadwaj and John S. Baras, University of Maryland

Abstract

The demand for Internet bandwidth has grown rapidly in the past few years. A new generation of broadband satellite constellations promise to provide high speed Internet connectivity to areas not served by optical fiber, cable or other high speed terrestrial connections. However, using satellite links to supply high bandwidth has been difficult due to problems with inefficient performance of the Internet's TCP/IP protocol suite over satellite. We describe an architecture for improving the performance of TCP/IP protocols over heterogeneous network environments, especially networks containing satellite links. The end-to-end connection is split into segments, and the protocol on the satellite segment is optimized for the satellite link characteristics. TCP congestion control mechanisms are maintained on each segment, with some coupling between the segments to produce the effect of end-to-end TCP flow control. We have implemented this design and present results showing that using such gateways can improve throughput for individual connections by a large factor over paths containing a satellite link.

Keywords: Satellite, TCP/IP, TCP, Gateway, Broadband, Geostationary, GEO, Network, Internet, Ka, Latency

1 Introduction

In recent years the global Internet has experienced unprecedented growth. However, this growth has largely been confined to areas where a good cable or optical fiber based communications infrastructure exists. For many geographically remote or underdeveloped regions, creating such an infrastructure is time-consuming and expensive. Satellite links, on the other hand, can be set up relatively quickly, and coverage can be extended to large regions at high data rates.

A new generation of broadband satellite constellations will begin deployment early in the next decade. These satellites will have onboard switching capabilities and hundreds of spot beams, and they will tap previously unused spectrum in the wide-band Ka region. Thus satellites present an ever more attractive solution for providing information access to remote areas, and will become a viable alternative to terrestrial connections even in high density areas. In military systems, where instant connectivity to any destination is an important requirement, satellite links are virtually a necessity.

The most cost-effective solution for satellite coverage is the Goestationary Earth Orbit (GEO) whereby a satellite is "parked" at a fixed location about 22,300 miles above the equator. The speed of light dictates a latency of approximately 250 milliseconds on a typical "hop" comprising a GEO satellite uplink and downlink. This latency as well as bit errors present in satellite links have some adverse effects on the Internet Protocol (IP) suite of data communication protocols used in the Internet.

The IP suite of protocols forms the basis of the Internet today. The Internet Protocol (IP) [1] provides the functionality for routing packets of data (known as datagrams) from source to destination. Packets are routed based on the source and destination addresses in their IP headers.

A number of protocols are built on top of IP to provide transport layer functionality. The most important are the User Datagram Protocol (UDP) [2] which deals with unreliable data delivery between endpoints, and the Transmission Control Protocol (TCP) [3, 4] which provides for reliable delivery. UDP includes a checksum in its header so that arriving datagrams can be checked for corruption, however it does not by itself provide any means of recovering datagrams that are corrupted, misdelivered or lost in the network. TCP uses a system of sequence numbers and acknowledgments to ensure reliable delivery. Thus TCP, unlike UDP, has a closed feedback loop. The presence of

this feedback loop causes TCP to be affected by issues like link delay. Both protocols are affected by bit errors, however the effects on TCP are far greater.

Examples of applications using UDP are streaming audio, streaming video, voice over IP and desktop video conferencing. On the other hand, telnet, FTP, email services and web browsers use TCP. In addition, many web-based audio-video services use a combination of UDP and TCP. Thus a large (and growing) fraction of traffic in the present-day Internet is TCP traffic. The remainder of this paper will address issues of TCP over GEO satellite.

The performance of TCP over satellite can be somewhat improved by using performance-enhancing TCP options [5]. However, these options are not available in all TCP implementations. Due to the large installed base of legacy TCP/IP stacks, it seems unlikely that they will be widely used in the immediate future. This paper discusses an architecture for seamless integration of satellites into existing TCP/IP networks. The architecture described here is general, and can be generalized to any transport protocol over any heterogeneous network. The work described herein is part of an ongoing project to provide efficient TCP/IP performance over geostationary satellite links which may have onboard switching capabilities.

2 Limitations of TCP over satellite

In this section we look at some factors that limit TCP performance over satellite links and discuss some of the mitigations that have been proposed to reduce their impact.

2.1 Effect of high bandwidth-delay product

TCP uses an algorithm similar to selective repeat ARQ to provide reliable delivery of data. The header of each segment contains an offered window, which represents the largest amount of data that the host permits the remote end to send without receiving further permission. This offered window is represented in the header by a 16-bit field, which restricts its value to 64 kilobytes. Some implementations limit the maximum window size to 32 kilobytes, and many popular implementations default to a window of 8 kilobytes. Since TCP cannot send more than one window of data per round-trip time, the maximum throughput attainable by a connection over a geostationary satellite link is restricted to 128 kbps[1]. The Window Scaling option [6] allows the effective size of the offered window to be increased to 30 bits, but many network applications need to be modified to be able to use the larger window sizes.

2.2 Effect of high delay on congestion control mechanisms

TCP uses a closed-loop positive feedback mechanism to determine its rate of transmission. To avoid congestion in the network, every connection starts by sending data at the lowest possible rate and increases its rate of transmission as acknowledgments are received for the data sent. The rate of transmission is governed by the congestion window, and this window is grown is in two phases. In the first phase, known as slow start [7], the congestion window is initialized to one segment (resulting in a transmission rate of one segment per round trip time), and is increased by one segment for every new acknowledgment received. When the congestion window passes a threshold, the congestion avoidance phase is started, and the window is increased by one segment every time a complete window of data is acknowledged.

Due to the low initial window in slow start, a significant number of round trips may be required for the congestion window to grow large enough to effectively utilize the link bandwidth. This is a problem in the satellite environment, where the round trip delays are long. The Internet Engineering Task Force (IETF) has approved as experimental a proposal [8] to increase the initial value of the congestion window to four packets instead of one. Some preliminary work suggests that this does not significantly degrade performance even in low-bandwidth situations [9, 10].

A different problem is seen during the congestion avoidance phase. In this phase the window grows by only one segment every round trip time, and so window growth is much slower than in slow start. Thus, if the congestion window is too small when congestion avoidance is entered, the satellite link can be under-utilized for prolonged periods of time. Most data transfers over a data link can thus complete without ever having attained a window large enough for optimal link utilization.

[1] Assuming a window size of 8 kB and a round-trip delay of 500 ms over the satellite link.

2.3 Effect of bit errors

Raw satellite links are more noisy than cable or fiber links. Bit error rates of the order of 10^{-6} are often observed, even under good conditions. Most present-day satellite systems use Forward Error Correction techniques to reduce the bit error rate to less than 10^{-10}. However, in some environments, such as in military systems, higher error rates can be present. Further, errors on satellite links tend to be bursty by nature.

TCP assumes that all packet losses are due to congestion, or, in effect, that there are no bit errors in the network. When a packet loss is encountered, the lost packet is retransmitted and the rate of sending is reduced (at least halved, see [7]). This leads to many problems on links with measurable bit error rates. When a packet is lost due to bit errors, the sender's congestion window is halved even though no congestion is present, thus under-utilizing the link. In addition, TCP uses a cumulative acknowledgment scheme, and so can discover only one segment loss every round trip. Thus if multiple segments are lost in one window of data, throughput is reduced sharply. This second problem can be partially solved by the use of the Selective Acknowledgment option [11].

2.4 Effect of heterogeneous network topology

TCP uses a self-clocking mechanism - the acknowledgments provide the clock by which the sender paces itself. This leads to unfairness between connections that traverse widely differing paths in the network - connections with smaller round-trip times can increase their rate of sending more rapidly, and so end up capturing most of the network bandwidth, at the expense of long-delay connections [12].

3 Gateway Design

We now describe an architecture that attempts to solve the problems enumerated above in a way that is transparent to the network user. It does not require any changes, including configuration changes, to end-user terminals or TCP/IP stacks. The only changes are made to the gateways on either end of the satellite link. The goal is to preserve the end-to-end semantics of TCP, while ensuring fairness and optimal link utilization over variable-bandwidth satellite links.

3.1 Connection splitting

Our approach uses connection splitting, as shown in Figure 1. The end-to-end TCP connection between H1 and H2 is broken into three TCP connections, namely C1, C2 and C3, by the gateways G1 and G2. On each of these "connection segments", TCP is used for reliable delivery. However, the TCP used on C2 is enhanced with the timestamp and window scaling options [6] as well as the SACK option [11]. It uses the FACK [13] congestion control algorithm, and an increased value for the initial congestion window during connection startup. Queue management policies and flow control policies at the gateways provide feedback between the connection segments, thus producing the effect of end-to-end TCP flow and congestion control.

All our modifications are carried out at the TCP and IP layers, and the implementation is independent of the underlying link layer. In particular, no effort is made to make the link layer TCP-aware. This avoids the possibility of undesirable interactions between the link layer and the TCP congestion control mechanisms. Within the TCP layer, the basic flow and congestion control mechanisms are preserved.

The procedure used to perform connection splitting is as follows. Whenever a gateway sees a connection request (i.e. a SYN segment), it intercepts the request and originates a similar connection request with an enhanced option set. When all downstream connections are completed, an acknowledgment (i.e. a SYN ACK) is returned to the host that originated the original request. Both the gateways always negotiate and accept all the TCP options listed above during connection setup, so that the connection between G1 and G2 will always use all these options.

Once the connection has been set up, the gateway intercepts all data on that connection, returns an acknowledgment to the sender bearing the address of the destination, and buffers the data for downstream transmission. A queue management algorithm similar to RED is used at the input to keep gateway queues small. In addition, a flow control

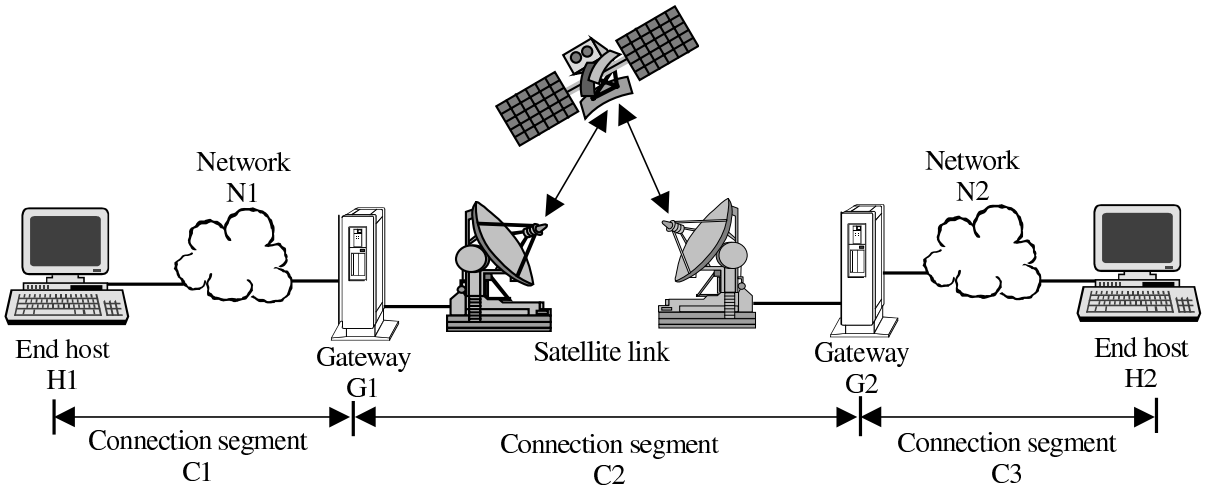

Figure 1: Overview of TCP connection splitting.

algorithm is used to pass information about downstream congestion to the queue management function and hence to the upstream connection. The flow control implements a "back-pressure" algorithm. Incoming segments on the upstream connection are serviced at a rate that matches the rate of transmission on the downstream connection. Thus when the downstream path gets congested, the offered window on the upstream connection is reduced correspondingly, and congestion information is propagated back to the sender.

When a gateway receives a FIN segment, it immediately closes the corresponding half-duplex connections. When a FIN has been received for both directions of a TCP connection, all the resources for the corresponding split connection segments are freed to minimize resource usage.

A timing diagram of a simple data transfer is shown in Figure 2. Hosts and gateways are named as in Figure 1. Note that there is a variable delay due to buffering at G1 as well as G2, which is not shown in the figure. Also, a host may choose to piggyback ACKs on other kinds of packets. In the figure, AckH1 may be piggybacked on the data following it, and FinH2 may be combined with the ACK immediately preceding it.

The following steps are taken to preserve transparency and end-to-end semantics:

- When a gateway receives a SYN for a connection, it does not return a SYN ACK until it receives a SYN ACK from the downstream node.
- All IP addresses, port numbers and sequence numbers are preserved throughout.
- No header information is changed or added at any stage except for TCP options, which may be different on each connection segment. No encapsulation or tunneling is performed.
- A fallback mode is used when resources are not available to set up a connection or when a packet is received on an unknown TCP connection. In these cases the packet is forwarded at the IP layer. This, along with the preservation of sequence numbers, provides a measure of robustness with respect to routing changes in the network.
- All non-TCP packets, as well as TCP packets using IPSEC or other mechanisms to hide the TCP header, are forwarded at the IP layer. Thus TCP connections using IPSEC go through as usual, but do not benefit from the gateway enhancements.

3.2 Discussion of connection splitting

The concept of connection splitting can be readily extended to more complicated networks. Instead of running TCP on all the connection segments in our setup, we could, with slight change in the mechanisms for feedback between

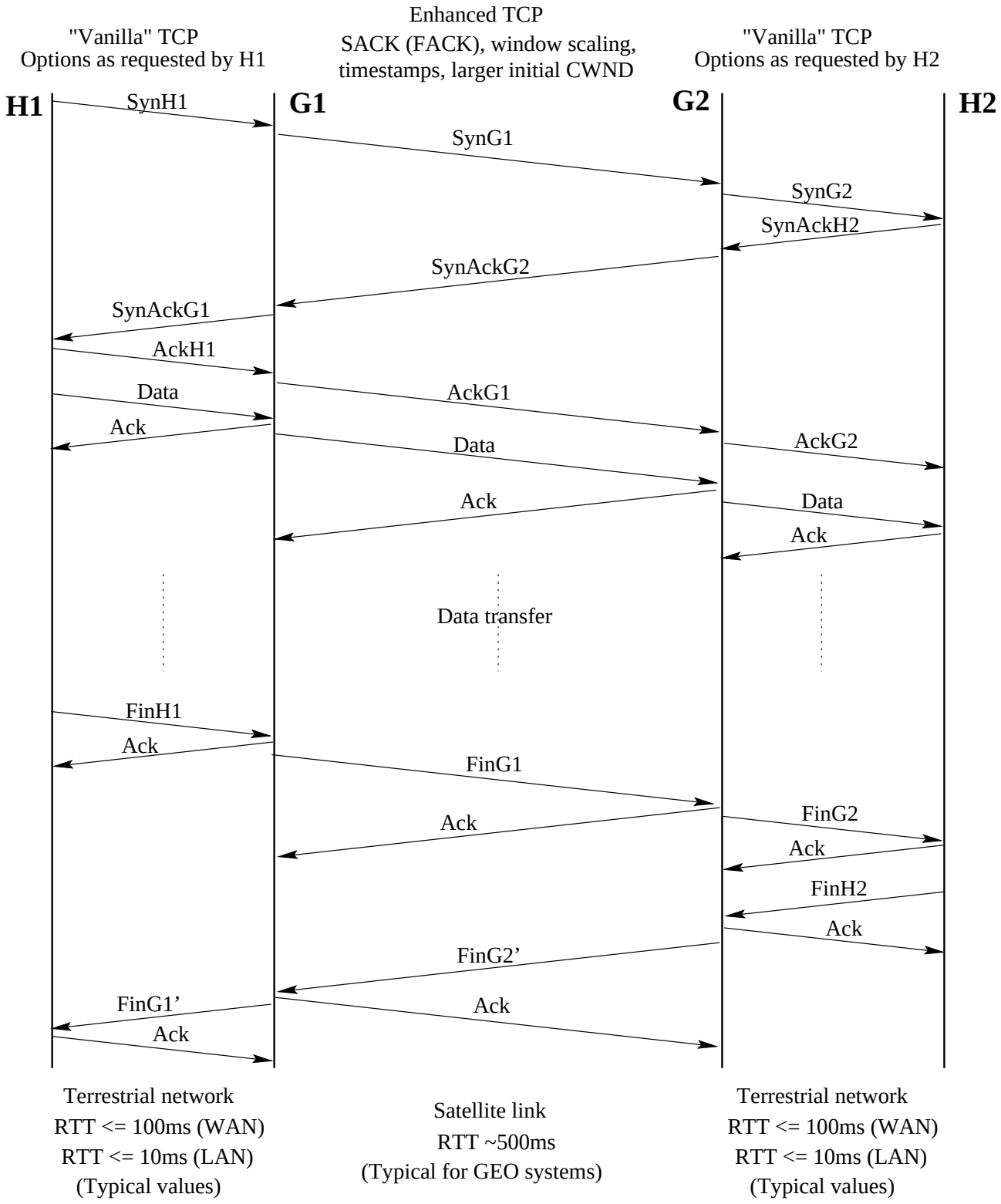

Figure 2: Timing diagram for simple unidirectional transfer from H1 to H2.

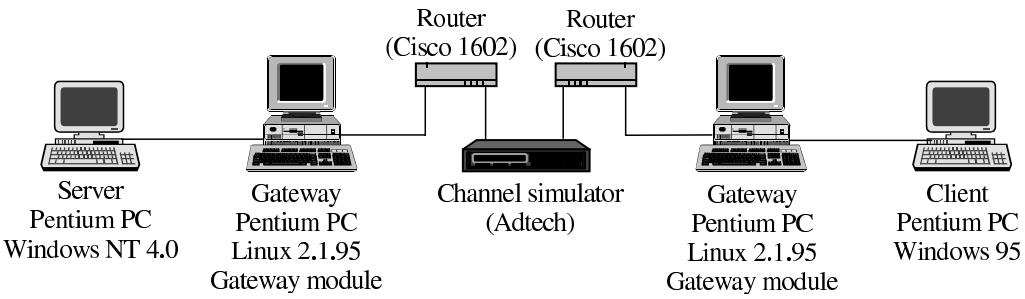

Figure 3: Exact test configuration.

segments, adapt the gateways to translate between arbitrary protocols. Also, due to the transparent nature of the gateways, there is no constraint on the number of gateways in a path. Thus, if a terrestrial wireless link was present in the network, transparent gateways could be placed on either end of this link as well. These gateways could use a protocol optimized for terrestrial wireless links on the segment between them, while retaining interoperability with the rest of the network.

The use of connection splitting alleviates the problem of TCP unfairness to connections with long round-trip times. Connection segments C1 and C3 (see Figure 1) experience round-trip delays that are of the same order as those seen by other users of networks N1 and N2 respectively, and so do not have to pay a penalty for using the satellite link. Thus they find themselves able to compete fairly with other users of N1 and N2 respectively. On the satellite link (segment C2), all the connections compete over the same round-trip times, and so no connection can capture more than its fair share of bandwidth[2]. Thus by using gateways to partition the network into nearly homogeneous subnetworks, good performance can be obtained without the need to resort to more complex end-to-end protocols.

4 Performance measurements and analysis

Our current implementation of the transparent TCP gateway functionality runs on the Linux operating system, kernel version 2.1.95. Though the gateway module is processor-independent, all the tests have been carried out on a pair of Pentium PCs running at 166 MHz. The test configuration is shown in Figure 3. The server was running Microsoft Windows NT Workstation 4.0 with the default TCP/IP parameters, while the client was running Microsoft Windows 95 with the default parameters. The server used was running the NT Peer Web Services software, comprising an FTP server and an HTTP server. The client for the FTP testing was the standard FTP client supplied with Windows 95. A data channel simulator was used to simulate the satellite channel.

We measured throughput for single FTP connections using different file sizes, with different data rates, delays and error rates on the simulated satellite link. There was no other traffic on the link. Files of sizes 10 KB, 100 KB, 1000 KB, 10000 KB and 100000 KB were tested. The link rates used were 384 kbps, 1.536 Mbps and 8 Mbps. Tests were carried out for zero delay and for a delay of 250 ms each way on the link. These delays were chosen to approximate a terrestrial leased line and a typical geostationary satellite link respectively. In each of the above cases, we measured throughput for bit error rates of 0, 10^{-9}, 10^{-8}, 10^{-7} and 10^{-6}. To provide a baseline for comparison, identical tests were carried out with the gateway machines functioning as IP routers without any connection splitting. The observed throughput was normalized it by the maximum possible throughput to obtain the efficiency. The maximum possible throughput was taken to be the fraction of link bandwidth usable for data after adjusting for TCP, IP and link layer headers.

Figure 4 shows the efficiency obtained for a 10 Megabyte file transfer over an error-free link with delay of 250 ms in each direction. As expected, the end-to-end TCP transfer is limited to a constant maximum transfer rate, independent of link bandwidth, by the window sizes and the link delay. Thus its efficiency decreases with increasing link bandwidth. The connection splitting approach, on the other hand, yields much better efficiency, relatively independent of the link bandwidth.

[2]Here we define fairness relative to TCP. Thus a bandwidth allocation is considered fair if it matches an allocation obtainable for equally competing TCP connections with similar parameters.

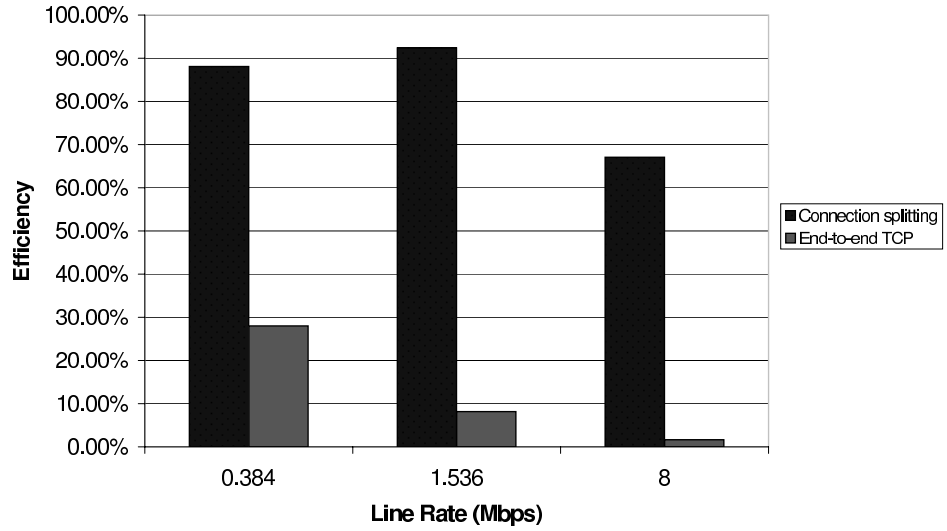

Figure 4: Utilization at different data rates.
Error-free link, Delay=250 ms each way, File size=10 Megabytes

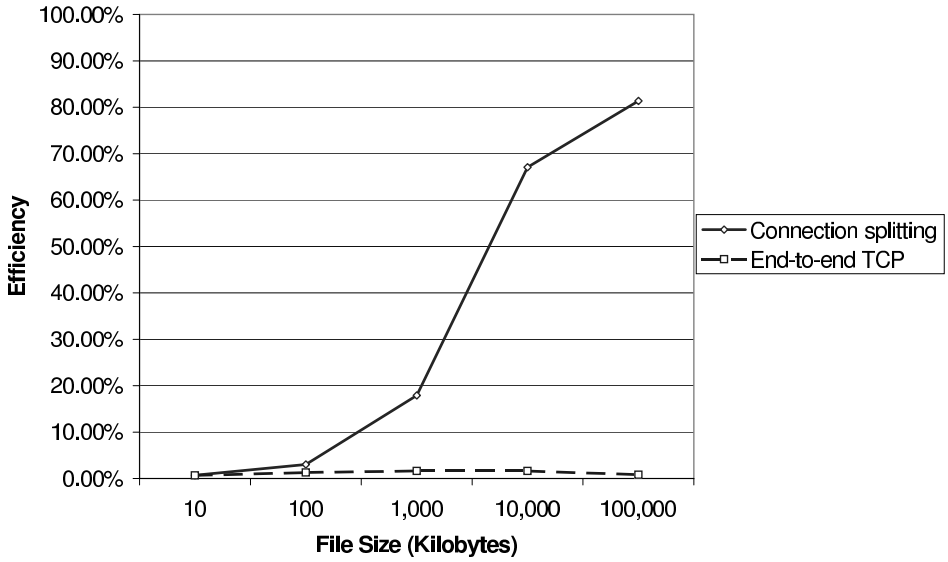

Figure 5: Performance comparison: effect of file size.
Error-free link, Delay=250 ms each way, Line rate=8 Mbps

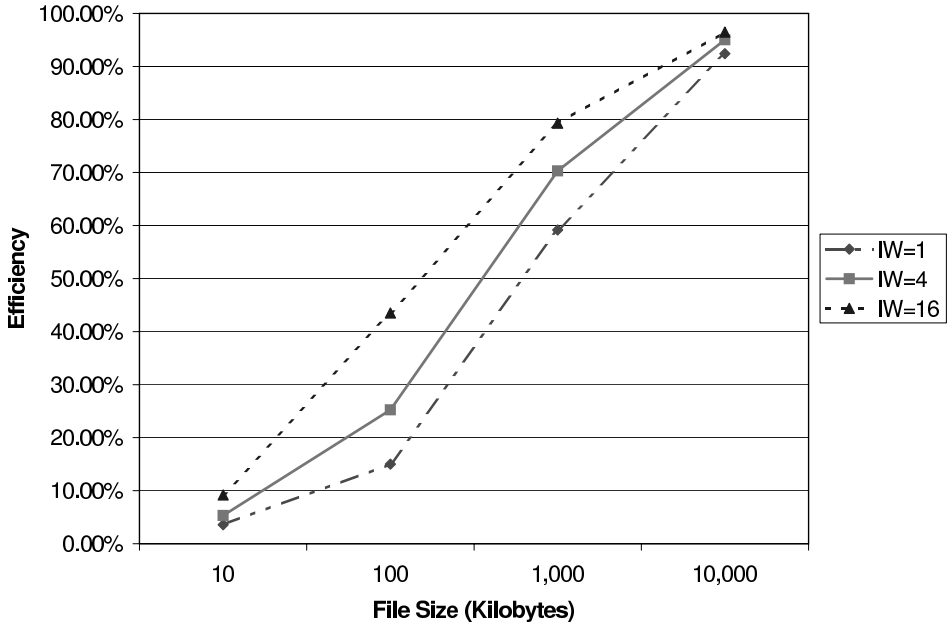

Figure 6: Gateway performance with different initial congestion window (IW) sizes.
Error-free link, Line rate=1.544 Mbps, Delay=250 ms each way

The lower efficiency for the split-connection case at the higher data rate was mainly due to slow start. With smaller file sizes, the TCP on the satellite link does not attain a large enough congestion window to achieve a rate of transmission sufficiently close to the link rate, and lower efficiency is observed. As the file size is increased, the cost of low utilization during slow start is amortized over a longer interval of near-line-rate transfer, so the utilization improves. Thus efficiency for small files is limited by slow start in the connection splitting case, whereas it is limited by window size in the end-to-end TCP case.

Figure 5 illustrates this effect for a line rate of 8 Mbps and a one-way link delay of 250 ms. As might be expected, the efficiency with connection splitting improves when the file size is an order of magnitude greater than the bandwidth-delay product of the link. Thus retaining the TCP congestion control mechanisms carries a penalty for small transfers. We are currently investigating methods to make TCP congestion control more aggressive without sacrificing its fairness and ability to adapt to changing network conditions.

One method of reducing the impact of slow start for small file transfers is to increase the size of the initial congestion window used at connection startup. We repeated our file transfer tests for the connection splitting case with the gateways set to use different values of the initial congestion window on the satellite link. As Figure 6 shows, using larger values increases the throughput for small files by a factor of two or three. For very small files (e.g. the 10 kB file in the graph) the actual throughput attained is still low because of the overhead due to the three-way handshake for connection setup. For very large files, the increase in the initial window does not have much effect since most of the transfer is done after the window has grown large enough to utilize most of the link capacity.

We now turn to the effect of bit errors on our gateway implementation. Figure 7 shows the effect of bit errors on file transfers over an end-to-end TCP connection. Our gateways use the FACK algorithm, which utilizes the information received in SACK (Selective ACKnowledgment) blocks to retransmit lost packets. As seen in Figure 8, this reduces the effect of bit errors on throughput, even at error rates as high as 10^{-6}. However, since the susceptibility of TCP to errors depends on the number of errors per round trip time, throughput is more affected at higher link rates. Throughput is relatively unaffected for small files, which are less likely to encounter any bit errors in transmission, whereas large files are affected considerably.

Figure 9 shows the variation of efficiency with bit error rate for our gateway implementation. Again, high-bandwidth links are more affected, and performance drops sharply when the error rate is close to one error per round trip. Thus we see efficiency drop considerably at a line rate of 1.536 Mbps when the error rate is 10^{-6}, and the efficiency at

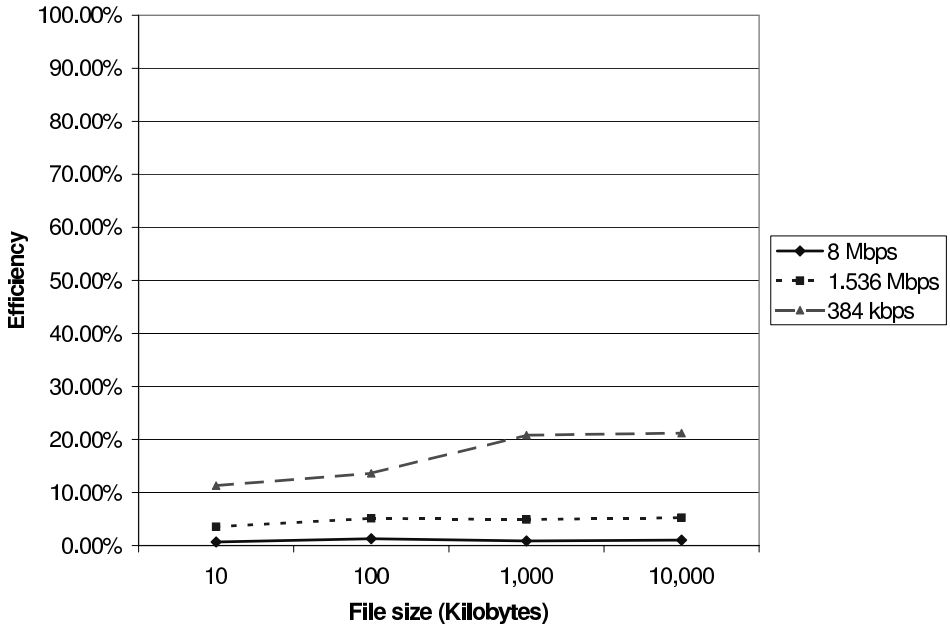

Figure 7: Performance with delay and bit errors: End-to-end TCP.
Delay=250 ms each way, BER=10^{-6}

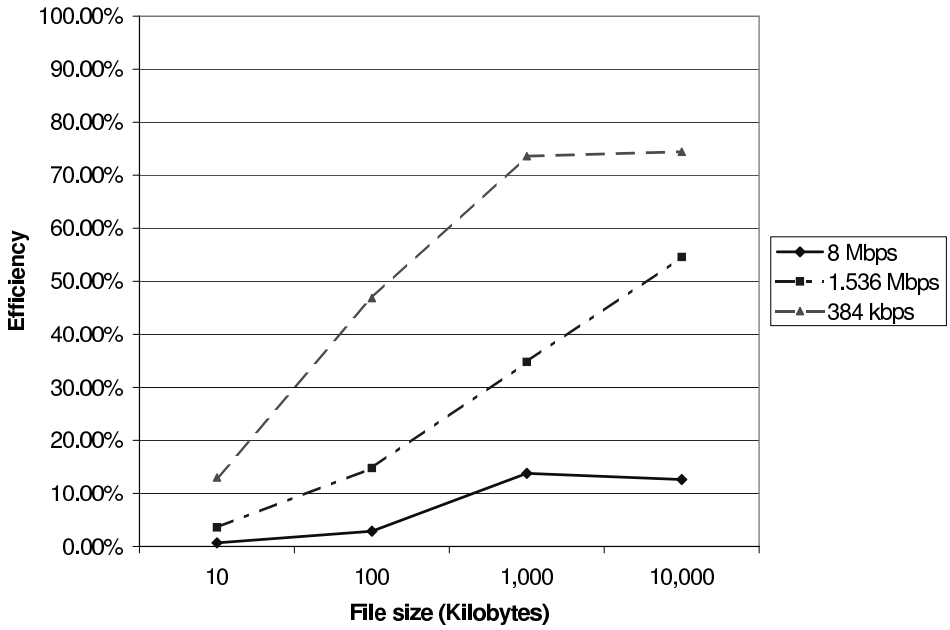

Figure 8: Performance with delay and bit errors: Connection splitting case.
Delay=250 ms each way, BER=10^{-6}

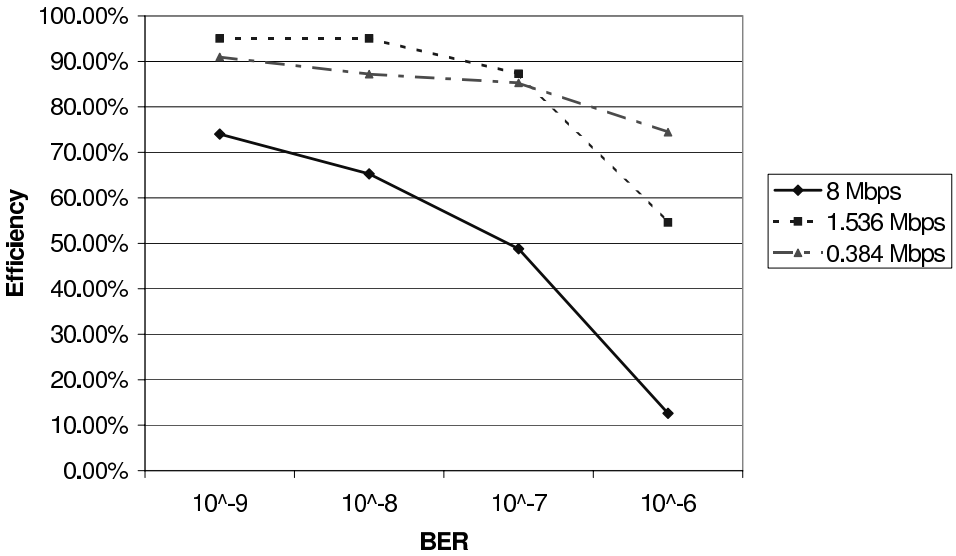

Figure 9: Performance with different BER.
File size=10 Megabytes, Delay=250 ms each way

8 Mbps drops below 50% when the error rate reaches 10^{-7}.

In addition to the above tests with FTP, we also carried out similar tests with HTTP transfers. The server was the HTTP server included with Microsoft Windows NT Workstation 4.0, and the client was Netscape Communicator, version 4.05. This HTTP client uses a single persistent TCP connection to the server to transfer all the objects. Thus it avoids the overhead of a three-way handshake for each object on a webpage. Tests were carried out with the gateways acting as IP routers and with the gateways performing connection splitting. We show the results of two such tests in Figure 10 and Figure 11. In the case shown in Figure 10, the webpage consisted of a single large image of size 391 KB, embedded in an HTML document of size 356 bytes. In the tests depicted in Figure 11, the webpage was composed of an HTML document of size 1.7 KB, and 16 GIF and JPEG images. The image sizes ranged from 19 KB to 100 KB, and the total size of the images was 669 KB.

HTTP uses a request-response mechanism, wherein the client requests one object at a time from the server. Thus there is an interval of one round trip between the time that the client finishes receiving an object and the time that it begins to receive the next object. During this time, there is no traffic on the link except for the request by the client. Therefore it is meaningless to compute the efficiency for such transfers, as the traffic generated is intermittent in nature, with long pauses. We have plotted the total time required for each webpage to load, as measured by a stopwatch. As seen in the graphs, there is still a considerable improvement when connection splitting is used. However, due to the pauses during HTTP requests, the improvement is not as large as that seen with FTP.

5 Conclusions and future work

In this paper we demonstrated an architecture for a transparent gateway which uses connection splitting to enhance TCP performance over satellite links. Such gateways are easy to incorporate into existing networks, and improve the performance of existing TCP implementations by a large factor in the presence of satellite links. The proposed architecture also allows for end-to-end congestion control and fair allocation of bandwidth among competing network flows.

Throughput on the satellite link is affected substantially by the TCP congestion control mechanisms. We are currently investigating changes to the slow start and congestion avoidance algorithms and their effect on protocol fairness. Buffer management policies at the gateways, and feedback strategies for carrying out end-to-end flow control in our framework are also under investigation.

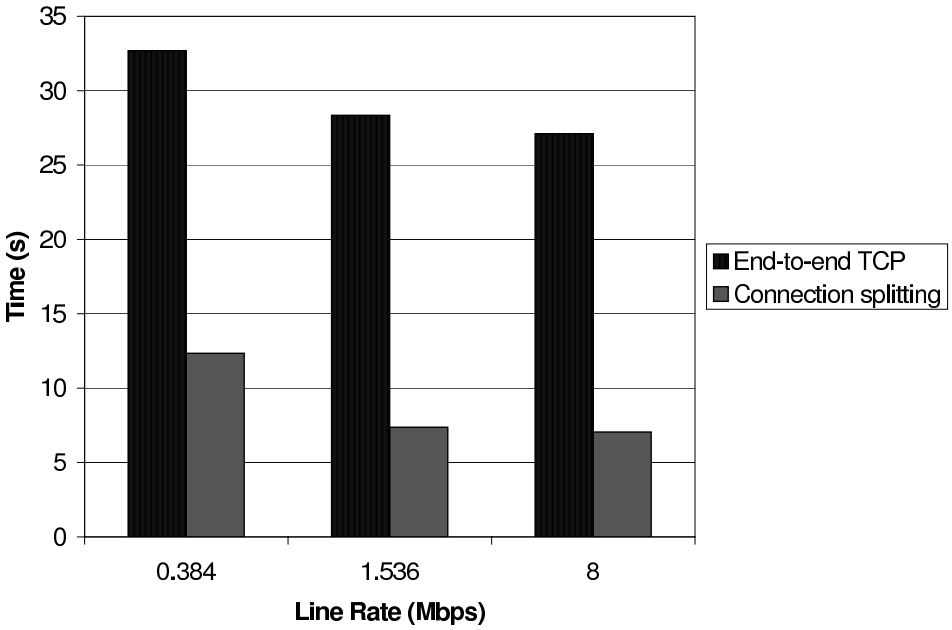

Figure 10: HTTP performance comparison.
Error-free link, Delay=250 ms each way
Webpage composition: 356 bytes HTML document + 391 KB image

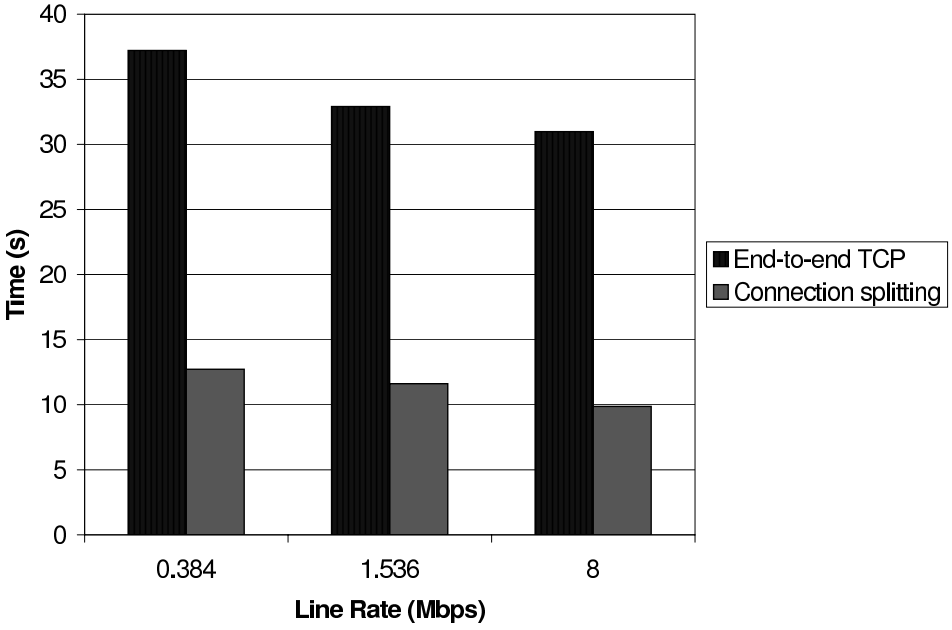

Figure 11: HTTP performance comparison.
Error-free link, Delay=250 ms each way
Webpage composition: 1.7 KB HTML document + 16 images (total 669 KB)

Acknowledgments

The authors would like to thank Rohit Tripathi and Koroush Saraf for their invaluable assistance in carrying out the performance tests described in Section 4. We are also indebted to Rohit Goyal for his help with the testing as well as many insightful comments and discussions.

References

[1] J. Postel ,ed. Internet Protocol – protocol specification. *RFC 791*, September 1981.

[2] J. Postel, ed. User Datagram Protocol. *RFC 768*, September 1980.

[3] J. Postel, ed. Transmission Control Protocol – protocol specification. *RFC 793*, September 1981.

[4] B. Braden, ed. Requirements for Internet hosts – communication layers. *RFC 1122*, October 1989.

[5] M. Allman and D. Glover. Enhancing TCP over satellite channels using standard mechanisms. Internet Draft draft-ietf-tcpsat-stand-mech-05.txt, August 1998.

[6] R. Braden V. Jacobson and D. Borman. TCP extensions for high performance. *RFC 1323*, May 1992.

[7] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *RFC 2001*, January 1997.

[8] Mark Allman, Sally Floyd, and Craig Partridge. Increasing TCP's initial window. RFC 2414, September 1998.

[9] K. Poduri and K. Nichols. Simulation studies of increased initial TCP window size. RFC 2415, September 1998.

[10] Tim Shepard and Craig Partridge. When TCP starts up with four packets into only three buffers. RFC 2416, September 1998.

[11] S. Floyd M. Mathis, J. Mahdavi and A. Romanow. TCP selective acknowledgment options. *RFC 2018*, October 1996.

[12] T.V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, June 1997.

[13] M. Mathis and J. Mahdavi. Forward Acknowledgment: Refining TCP congestion control. In *Proceedings of SIGCOMM '96*, Stanford, CA, August 1996.