TECHNICAL RESEARCH REPORT

Flow Control at Satellite Gateways

by Xiaoming Zhou, Xicheng Liu, John S. Baras

CSHCN TR 2002-19
(ISR TR 2002-37)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

Flow control at satellite gateways

Xiaoming Zhou, Xicheng Liu, and John S. Baras
Institute of system research
Center for satellite and hybrid communication networks
University of Maryland at College Park, MD, 20742

Abstract-- **Broadcast satellite networks are going to play an important role in the global information infrastructure. Several systems including DirecWay from Hughes Network System use satellites to provide direct-to-user high speed Internet services. TCP works well in the terrestrial fiber networks but does not work well in satellite hybrid (satellite-terrestrial) networks. In this paper we analyze the problems that cause this dramatically degraded performance. Based on the observation that it is difficult for an end-to-end solution to solve these problems in this kind of hybrid networks, we propose a connection splitting based solution. A rate based protocol is designed for the satellite connections and a flow control scheme at the satellite gateways (SGW) is used to couple the two split connections together. Our simulation results shows that our scheme can maintain high utilization of the satellite link and improve fairness among the competing connections.**

I. Introduction

For the home users or small enterprise, using dial-up modem to access the Internet is too slow. In order to provide broadband Internet service for these customers, satellite hybrid network was proposed to solve this last-mile problem (figure 1). This kind of hybrid networks exploits three observations [1]: 1) some rural areas may not be reached by fiber networks or it may be too expensive to do so; 2) satellite hybrid networks can provide high bandwidth to a large geographical area and it is easy to deploy; 3) home users usually consume much more data than they generate. So this asymmetric hybrid network fits in the need very well. The satellites are just signal repeaters in the sky and are usually called bent pipe satellites. They are layer one devices and no switching is performed on board.

A geo-synchronous orbit (GEO) satellite is about 36,000km above the earth. The propagation delay between the ground terminals and the satellite is about 125ms. Therefore a typical round trip time (RTT) for this system is about 580ms including about 80ms RTT for the terrestrial networks. The time taken by TCP slow start to reach the satellite bandwidth (SatBW) is about $RTT*\log_2(SatBW*RTT)$ when every TCP segment is acknowledged[2,14]. For a connection with large RTT, it spends a long time in slow start before reaching the available bandwidth. For short transfers, they could be finished in slow start, which obviously does not use the bandwidth efficiently. Some researchers propose to use a larger initial widow [3] up to roughly 4K bytes rather than one MSS[1] for slow start. So files less than 4K bytes can finish their transfers in one RTT rather than 2 or 3. Another proposal [4] is to cancel the delayed acknowledgement mechanism in the slow start so every packet is acknowledged and the sender can increase its congestion window (CWND) more quickly. For bulk transfers, TCP throughput is inverse proportional to RTT [5]. So TCP connections with larger RTTs do not get their fair share of the bandwidth when they compete with the connections with smaller RTTs. Using simulations, Henderson claims the 'Constant-rate' additive increase policy can correct the bias against connections with long RTTs [6]. However it is difficult to implement this policy in a heterogeneous network.

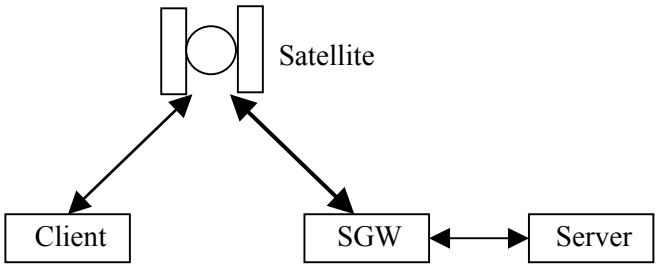

Figure 1 Direct to user satellite hybrid network

The bandwidth delay product in the satellite hybrid network is very large. In order to keep the large pipe full, the window should be at least the bandwidth delay product. However, the receiver advertised window that is 16 bits in the TCP header cannot be more than 64k bytes, which limits the two-way system throughput to 64k/580ms i.e. 903Kbps.Window scaling [7] is proposed to solve this problem. But when the window is large, it is more likely that multiple packets are lost in one window caused either by congestion or link layer corruptions or both. The multiple losses will trigger TCP congestion control algorithms and lead TCP to actually operate with

[1] Maximum segment size

a small average window. For the same reason, the sender buffer size can also limit the TCP connection throughput if it is less than the bandwidth delay product, which is usually the case in a lot of operating systems.

Ka band satellite channel is noisier than fiber channel. Bit error rates of the order of 10^{-6} are often observed [8]. Because TCP Reno treats all losses as congestion in the network, this kind of link layer corruptions can cause TCP to drop its window to a small size and leads to poor performance. TCP SACK[9] can convey non-contiguous segments received by the receiver in the acknowledgements (ACKs) so that the sender can recover error much faster than TCP Reno, which well known can recover only one loss per RTT. Forward error correction (FEC) coding is usually used in satellite communication to reduce the bit error rate. However, FEC consumes some bandwidth by sending redundant information together with the data and transforms the original random error nature to one with bursty errors.

In the satellite link layer, time division multiplex (TDM) is used for the downlink and multiple frequency time division multiple access (MF-TDMA) is used for the uplink. The downlink bandwidth from the satellite to the earth terminals is much larger than the uplink bandwidth. When the uplink traffic load is greater than the uplink bandwidth, congestion could happen. The congestion in uplink may cause poor performance in the downlink because TCP uses ACKs to clock out data. In the best case, the ACKs are not lost, but queued, waiting for available bandwidth. This has a direct consequence on the retransmission timer and slows down the dynamics of TCP window. In one way transfer, most of the time the uplink is transferring pure ACKs. To alleviate this problem, ACK filtering [10] was proposed to drop the ACKs in the front of the IP queue by taking advantage of the cumulative acknowledgement strategy in TCP. The situation is even worse for two-way transfers. When the users are sending data (say email with large attachment or upload file using FTP) and browsing the web at the same time, a lot of data packets could be queued in front of ACKs in a FIFO queue, which increases the ACKs delay dramatically. In this case, a priority queue can be used to schedule the ACK to be sent first [10].

In this paper, we present a connection splitting based solution [18,20,8,16] to the above problems in the satellite hybrid networks. A modified version of TCP with newly designed congestion control and error control algorithms is used for the satellite connections. A selective acknowledgement (SACK) based flow control scheme is used to couple the satellite connections and the terrestrial connections, which can maintain smoother flows with less buffer requirement at the satellite gateway than using TCP for both sides.

The rest of this paper is organized as follows. Section II provides the motivation for connection splitting schemes and describes the queuing model at the satellite gateway. Section III presents the congestion control and error control algorithms in the modified TCP for the satellite connections. Section IV presents the flow control algorithm at the satellite gateway. Section V gives the simulation results. Section VI relates our work to other proposed schemes for improving TCP over satellite links. Finally, Section VII concludes this paper.

II. Connection Splitting And Queuing Models

Satellite TCP connections need large windows to fully utilize the available bandwidth. However it takes much longer for satellite TCP connections than for terrestrial TCP connections to reach the target window size because of the large propagation delay and the slow start algorithm in TCP. And the window multiplicative decrease strategy makes the hard gained large TCP window very vulnerable to congestion. The misinterpretation of link layer corruption as congestion makes this situation even worse. In the best case, the packet loss does not cause timeout and TCP can stay in congestion avoidance phase rather than in slow start, the additive increase strategy makes the window to grow very slowly. From the above observations, we can see that even if the window scaling option is available, it is difficult for satellite TCP connections to actually operate with large windows. Therefore satellite connections cannot get their fair share of bandwidth when they compete with connections with smaller RTTs. It is difficult for end-to-end solutions to solve this fairness problem [6].

Because the feedback information of the satellite networks is either delayed too long or too noisy or both, end-to-end schemes cannot solve these problems very effectively. An alternative to end-to-end schemes is to keep the large window of packets in the network such as at the satellite gateway between the satellite and terrestrial networks. Considering the interoperability issue, we propose a connection splitting based scheme and design a flow control algorithm for the satellite gateway, which couples multiple terrestrial and satellite connections together to improve fairness among connections and to maintain high utilization of the satellite link. Basically the satellite gateway tries to hide the long propagation delay and link layer corruptions from the Internet servers.

In our scheme, an end-to-end TCP connection is split into two connections at the satellite gateway (figure 1). One connection is from the Internet server to the satellite gateway and another one is from the satellite gateway to the client. Observe that the users consume more data than they generate. We consider only the data transfer

from the Internet servers to the very small aperture terminal (VSAT) clients. Satellite gateway sends premature acknowledgements to the Internet servers and takes responsibility to relay all the acknowledged packets to the clients reliably. Although the Ka band satellite can provide higher bandwidth than Ku band satellite, satellite bandwidth is still a scarce resource compared to the bandwidth provided by optical fibers in the terrestrial networks. Therefore we assume the satellite link is the bottleneck of the system and the terrestrial networks have enough bandwidth. However our scheme still takes into account bottlenecked or idle terrestrial connections to achieve the efficient use of satellite link.

Based on the observation that the number of TCP connections is small at the client compared to that at the satellite gateway, we assume large buffer is available for each TCP connection at the client.

A. Queuing Model at the Satellite Gateway

For a normal router, only those packets waiting for transmitting are buffered at the IP layer. However, the satellite gateway has to buffer the packets waiting for transmission as well as those packets, which have been transmitted but not acknowledged. A normal router keeps all the packets in a FIFO queue while the satellite gateway has a queue for each TCP connection.

All the TCP packets received from the servers are forwarded to the TCP receive buffer of the SERVER-SGW connection and they are moved from the receive buffer to the send buffer of the SGW-CLIENT connection. Then the packets are sent from the send buffer to the clients over the satellite. From figure 2, we can see that the IP input queue should be empty if we assume the processing rate of the satellite gateway is not the bottleneck. The receive buffer and send buffer can be implemented by one physical buffer and data copy can be avoided by passing pointers. The queuing model at the satellite gateway can be simplified as in figure 2, in which the receive buffer and the send buffer are represented by one buffer.

The buffer size assigned to each connection at the satellite gateway has a direct impact on the end-to-end TCP throughput. Although memory is cheap, infinite buffer for each connection cannot be assumed because the satellite gateway is designed to support a large number of connections. If there is not enough memory available at the satellite gateway, newly arrived connections may have to be rejected or queued.

Firstly, assume there is only one connection in this system, the buffer size assigned to the TCP connection is

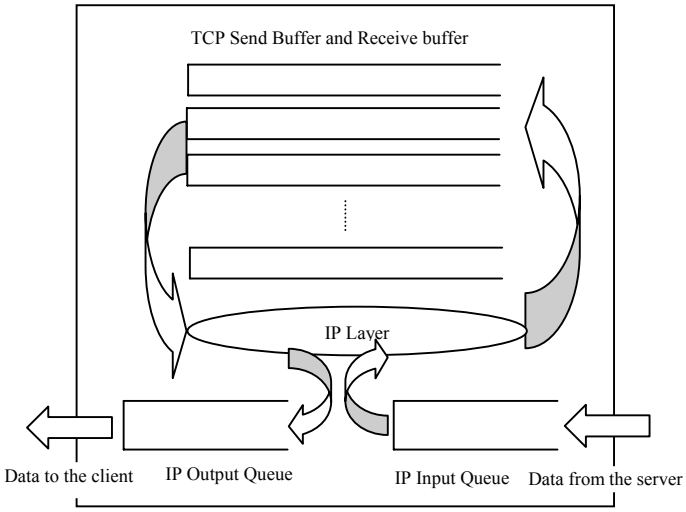

Figure 2 Simplified queuing model at satellite gateway

Buff and the effective satellite bandwidth[2] is SatBW. The data in the satellite pipe is SatWin[3] and the advertised receiver window for the server is RecvWin. The round trip time for the satellite connection is SatRTT and the round trip time for the terrestrial connection is TerrRTT. When the system reaches the steady state, the input rate of the queue at the SGW should be equal to the output rate of the queue, i.e. RecvWin/TerrRTT = SatWin/SatRTT. From [11], we know the throughput of the connection is MIN(SatBW, Buff/(SatRTT+TerrRTT)) and the backlog packets are MAX(0, Buff - SatBW* (SatRTT+TerrRTT)). From the above analysis and simulations results in figure 5 and figure 6, we can see that the buffer size can become the bottleneck of the end-to-end TCP performance if it is less than the bandwidth delay product. However when the buffer size is greater than the bandwidth delay product, there are packets backlogged at the satellite gateway and these backlogged packets cannot contribute to the throughput and only increase the queuing delay.

When there are multiple connections in this system, the bandwidth available to each connection is a function of the number of connections and their activities. One possible buffer allocation scheme is to use adaptive buffer sharing [12] to dynamically allocate a memory pool to TCP connections based on their bandwidth usage. While this scheme can dramatically decrease the buffer requirement, it does increase the implementation complexity. Based on our measurements at satellite gateway, although the mean of the number of active connections is large, the variance is small. Therefore the

[2] Effective satellite bandwidth is the raw satellite bandwidth deducted by the bandwidth consumed by the protocol headers.

[3] SatWin is neither congestion window nor the receiver advertised window. It is the number of packets in flight over the satellite link.

bandwidth available to each connection does not vary dramatically. We propose to assign each connection a static peak rate, which is the maximum bandwidth it can achieve and is much smaller than the total satellite bandwidth, and the buffer size is set corresponding to this peak rate. In practice, the peak rate can be set based on the measurements of the traffic characteristics and the target satellite link utilization.

III. Rate Based Reliable Protocol For Satellite Connections

TCP is a generic reliable protocol designed for wide area networks with optical channels in mind. Although TCP congestion control algorithms can guarantee network stability and fairness among TCP connections in terrestrial fiber networks, it is not efficient and effective in satellite networks.

Besides the inefficient congestion control, TCP windowing scheme ties the congestion control and error control together and errors can stop the window from sliding until they are recovered. The above observations motivate us to decouple the congestion control and error control in TCP first and then design more efficient and effective congestion and error schemes with our specific network characteristics in mind. Our goal is to maintain high utilization of the satellite link and to improve fairness among competing TCP connections. We chose to modify TCP to fit in the satellite networks rather than to design a new protocol from scratch because TCP is well understood and is the dominant transport layer protocol of the Internet.

A. Congestion Control

For the satellite connections, the satellite link bandwidth to be shared among them is fixed and known. Besides the number of connections and the traffic arrival pattern are known. All this information is available at the satellite gateway. Therefore there is no need to use slow start to probe the bandwidth and use additive increase and multiplicative decrease congestion avoidance to guarantee fair resource sharing as in the distributed case.

In our scheme, we cancel all the congestion control algorithms in TCP and substitute them with a scheduler as a centralized congestion manager. Also there is no need to exponentially back off the timer after timeout because congestion is taken care of by the scheduler i.e. congestion is impossible over the satellite link. Timer is used only for error recovery. The scheduler pulls the packets from the queues at the satellite gateway. When the scheduler encounters an empty queue, it goes on to serve the next one. As long as there are packets buffered at the satellite gateway, the satellite link can be fully utilized. Although weighted fair queuing (WFQ) can be used to provide fair sharing of the satellite bandwidth, we choose round robin as our scheduler because of its simplicity. Fairness is guaranteed by using the same maximum segment size for all the TCP connections. When the traffic load increases, the buffers begin to be filled up and the congestion is back pressured to the sources through the advertised receiver windows. When the traffic load decreases, the buffers begin to be emptied and larger advertised receiver windows are sent to the source so the sources can speed up. If some connections are bottlenecked upstream to the satellite gateway or are idle because the application layers do not have data to send, the scheduler can send packets from other connections. This way satellite link efficiency is achieved.

We assume large but not infinite buffer is available at the client and the TCP flow control is still enforced so that the open looped scheduler will not overflow the receiver's buffer. We do not use Window scaling to advertise large windows to the satellite gateway because large window scale factor can produce inaccurate values. In our scheme, the 16-bit receiver window field is still used but its unit is maximum segment size rather than byte. As long as the advertised receiver window allows, the packets are sent from the TCP layer to the IP layer and other packets are still buffered at the TCP layer. Only those packets at the IP layer can be sent by the round robin scheduler. The packets at the IP layer are just logical copies of the TCP layer packets with added IP headers. The packets are released from the buffer only when they are acknowledged by the clients. Essentially, the congestion control in our scheme is enforced at the IP layer rather than at the TCP layer.

B. Error Control

TCP depends on duplicate acknowledgements and timer for error control. Because out of order packet arrivals are possible in the wide area networks, the fast retransmit algorithm is triggered after three rather than one or two duplicate acknowledgements are received. The three duplicate acknowledgements requirement puts a high burden on the return channel bandwidth. The high bit error rate of the satellite link can cause multiple packet losses in one RTT and may lead to timeout. Furthermore the loss probability over the satellite link is determined totally by the bit error rate and packet size, so the retransmission packets can be corrupted as probable as original packets when the error rate is high [13]. When the retransmitted packets are lost, timer could be the only means for error recovery. However, timer has to be conservative and is usually set much larger than the round trip delay to make sure the packet

does leave the networks. These conservative loss detection and recovery schemes in TCP are not effective in satellite networks and should be enhanced.

In our scheme, we explore the specific characteristics of our network. Firstly, because the congestion control is taken care of by the scheduler, congestion is impossible for the satellite connections and any loss must be caused by the link layer corruption. So the error recovery scheme can operate independently with the congestion control scheme. Secondly, the satellite link is a FIFO channel and out of order packet arrivals are impossible. We design a scheme similar to the scheme in [13]. The in order delivery information is used for error detection. All sent packets including retransmission packets are sorted in the order they leave scheduler. We keep track of the right most packets in sequence space of all selectively acknowledged packets. Whenever an acknowledgment is received, we compare the current right most packet in the ACK with previous one in sequence space. If the sequence number does not advance, our scheme does nothing. While the sequence number does advance, our error recovery scheme is triggered. The first match of the current right most packet in the sorted list must have arrived at the client. If a packet before the right most packet in the sorted list is neither cumulatively acknowledged nor selectively acknowledged, our scheme assumes the packet is lost and retransmits it. This way, our scheme cannot only recover the first time losses but also the retransmission losses. The lost packets are tagged for retransmission and they are sent with higher priority than new packets. Timer is still used as the last resort for loss recovery. However the timer has a finer granularity. After timer expires, two copies of the lost packet are sent to increase redundancy.

However, when a packet does reach the client but all the acknowledgments for it are lost, our scheme can retransmit this packet unnecessarily. In our scheme, one acknowledgement can carry up to four SACK blocks. As long as the acknowledgements are not sent very infrequently, this situation should be rare.

IV. Flow Control At The Satellite Gateway

The strategy of connection splitting is to divide a system into two sub-systems and tries to find an optimal solution for each of these sub-systems. Simply putting two optimized sub-systems together does not necessarily give the optimal solution from the system perspective. This is because there are interactions between these sub-systems. In figure 3, we show this phenomenon between a satellite connection, which uses the reliable protocol in section III and a terrestrial connection, which uses regular TCP. In this simulation, the satellite link bandwidth is 600kbps and the terrestrial link bandwidth is 1.2Mbps (figure 1). The RTT of the satellite connection is 500ms and the RTT of the terrestrial connection is 80ms. The segment sizes are 512bytes. The buffer size at the SGW is 87 segments, which is about the bandwidth delay product. The bit error rate is 10^{-6} and the file size is 3M bytes.

When the system reaches equilibrium, the data in flight for the terrestrial connection is about 12 segments. While in figure 3, we see large oscillation around the equilibrium points. The reason for this is as follows. In TCP SACK, only segments cumulatively acknowledged are released from the retransmission buffer. Segments selectively acknowledged are still kept in the buffer because the TCP receiver may renege and discard the SACKed segments when it runs out of buffer. The buffer occupied by these SACKed segments can cause the SGW to advertise a smaller or even zero window to the Internet servers. This actually slows down or even stalls the servers. This corresponds to the period when the number of segments in flight falls below the equilibrium points in figure 3. After the error is actually recovered, the cumulative acknowledgement may cover a large number of packets. A large advertised window will be sent to the servers and cause the server to send a large burst (spikes in figure 3). This large burst may overflow the edge routers in the terrestrial networks and cause server TCP to drop its window. Now the terrestrial connection could cause starvation of the SGW queue and become the bottleneck.

Another point of view to this problem is as follows: during the error recovery phase, the cumulative acknowledgement does not advance and the TCP Internet server interprets this as sending rate slowing down on the satellite side which actually is not right and leads to stall. However after the error is recovered, it is equivalent to the sudden increase of the output rate of the SGW queue. Because we don't have infinite buffer at the SGW, the packets are not local and starvation happens when the scheduler has to wait for the new packets to arrive. ACK pacing [4], which advertises the suddenly increased available buffer in several packets, does not help too much here because it only tries to bring the system back to the equilibrium after the starvation has already happened while it cannot prevent starvation from happening. Increasing the buffer size at the SGW can help to improve the throughput because the large buffer size allows some packets backlogged at the SGW, which can be sent during the starvation. However when the error recovery of the satellite connection is slow, it will eventually stall the Internet server and may cause even larger oscillation. A flow control scheme is needed to couple these two connections to eliminate the stall-starvation cycles.

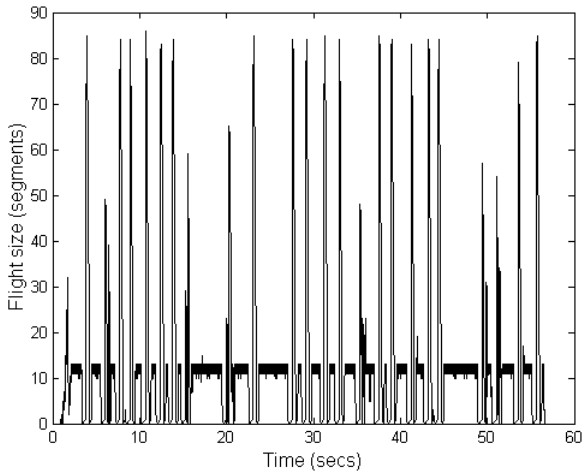

Figure 3 Data in flight of the server with normal SACK

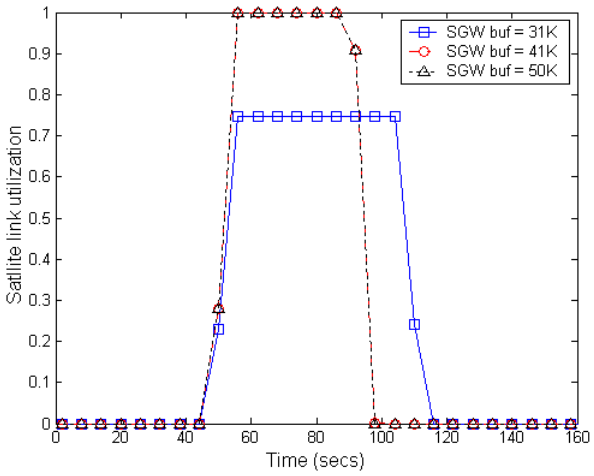

Figure 5 Satellite link utilization for different buffer sizes at satellite gateway

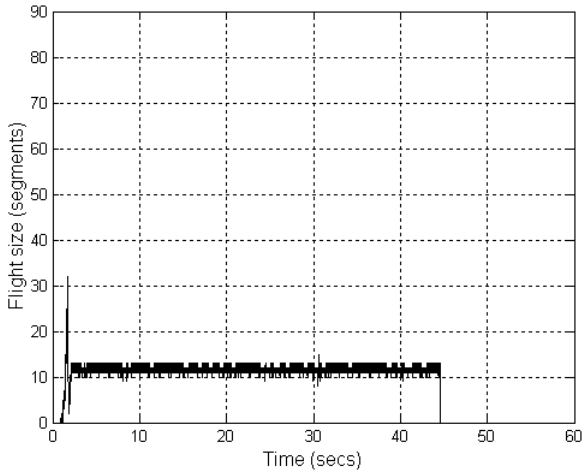

Figure 4 Data in flight of the server with modified SACK

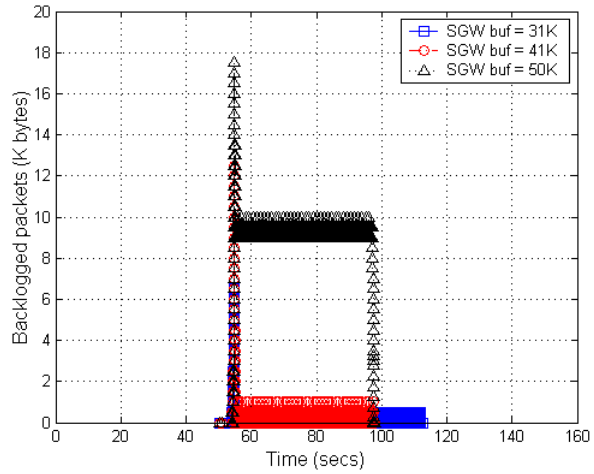

Figure 6 Backlogged packets for different buffer sizes at satellite gateway

The SACKed segments actually represent the output rate of the SGW queue. In our scheme, we change the SACK semantics. The TCP receiver never reneges and the TCP sender does not clear the SACK state information after timeouts. So the SACKed segments can be released from the retransmission buffer. Thus only those segments actually corrupted over the satellite link are still kept in the SGW buffer. The number of corrupted segments is much smaller than that of the SACKed segments. From figure 4, we can see our scheme can maintain a much smoother flow and can finish the transfer within a shorter time period.

Because of the uplink and downlink bandwidth asymmetry of the satellite channel, it is desirable to send fewer acknowledgements in the bandwidth limited return channel. However when fewer acknowledgements are sent, the number of segments acknowledged by each acknowledgement is increased. In order to maintain smooth data flow for the terrestrial connections, acknowledgements are paced out to the Internet servers based on the number of packets acknowledged by each acknowledgement and the acknowledgement inter-arrival time of the satellite connections. Although acknowledgements are not used to clock out data in our scheme, they are still used to recover the errors and to clear buffers. Less frequent acknowledgments could delay the error recovery and increase the buffer requirement at the SGW. Therefore there is a tradeoff between the return channel bandwidth requirement and the error recovery time as well as the buffer requirement.

V. Performance evaluation

In this section, we evaluate the performance of our scheme. The metrics we are interested in are satellite link utilization, end-to-end throughput and fairness.

A. Single connection case

Only one satellite connection and one terrestrial connection are set up for a bulk file transfer. The satellite link bandwidth is 600kbps and the terrestrial link bandwidth

is 1.2Mbps. The RTT of the satellite connection is 500ms and the RTT of the terrestrial connection is 80ms. The segment sizes are 512bytes and the file size is 3M bytes. Timer granularity is 100ms.

1) SGW buffer requirement and link utilization

In this section, the satellite link is set to be error free to get the buffer requirement at the SGW. When the buffer size at the SGW is set to 41K bytes, which is about the bandwidth delay product for the end-to-end connection, from figure 5 and figure 6 we can see that the satellite link is fully utilized and there are very few backlogged packets buffered at the SGW after the connection reaches the stable state. When the buffer size is decreased to 31K bytes, the connection becomes buffer bottlenecked and the satellite utilization is about 75%. This small buffer size limits the input rate of the SGW and there are no backlogged packets at SGW. However when the buffer size is increased to 50K, the connection becomes link bottlenecked and there are about 9K bytes data buffered at the SGW. The buffered data only increases the queuing delay at the SGW.

2) Return channel bandwidth requirement

The link rate in figure 7 is the raw satellite bandwidth deducted by the bandwidth consumed by the protocol headers and it is the upper limit of any achievable throughput. An acknowledgement is sent every N packets are received no matter they are in order or out of order. By changing N, we can change the acknowledgement frequency. Figure 8 shows that when the ACK frequency decreases exponentially, the return channel usage decreases exponentially. It is shown in figure 7 that the forward channel throughput is very insensitive to the return channel usage. Only when N increases up to 16, the forward channel throughput begins to decrease. This holds for both low bit error rate (i.e. BER = 10^{-6}) and high bit error rate (i.e. BER = 10^{-5}). Another interesting observation is that in order to get comparable forward channel throughput for higher bit error rate, more return channel bandwidth is required to provide timely information of the receiver buffer status.

B. Multiple connections case

Five servers communicate with five clients over the satellite link (figure 1). The raw satellite bandwidth is set to 3Mbps and the terrestrial bandwidth from each server to the SGW is 2Mbps. The RTTs for the five terrestrial connections are 2, 20, 40, 80 and 160ms respectively. The RTTs for all the five satellite connections are 500ms. The receiver buffer size at each client is 256K bytes. The buffer size for each connection at the SGW is set so that it can reach peak rate 900kbps.

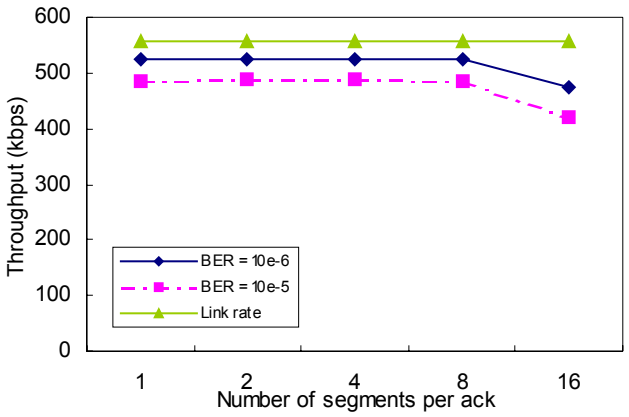

Figure 7 End-to-end throughput for different acknowledgement frequency

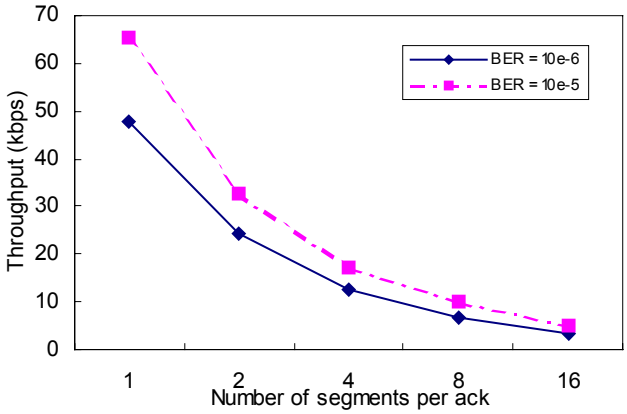

Figure 8 Return channel usage for different acknowledgement frequency

1) Throughput and fairness for bulk transfers

In this section, we use persistent traffic sources to test throughput and fairness. To focus on the forward channel performance, we choose the same acknowledgement frequency as in TCP, i.e. every other in sequence packet is acknowledged and every out of sequence packet is acknowledged. Figure 9 shows the aggregate throughput for the five transfers for our scheme and for TCP connection splitting scheme. The TCP connection splitting scheme uses TCP SACK for both the satellite connections and terrestrial connections. When the bit error rate is very low, both schemes can achieve very high throughput. For TCP connection splitting scheme when the bit error rate increases up to 10^{-6}, the link layer corruption causes the SGW TCP to drop its congestion window and the satellite connection occasionally stalls the terrestrial connection, which leads to degraded performance. When the loss rate is very heavy such as greater than 10^{-5}, the retransmitted packets can get lost again and TCP may have to wait for the timeout to recover the error. After timeout, the congestion window is set to one and TCP enters slow start. Therefore the satellite link utilization is very low for high loss rate.

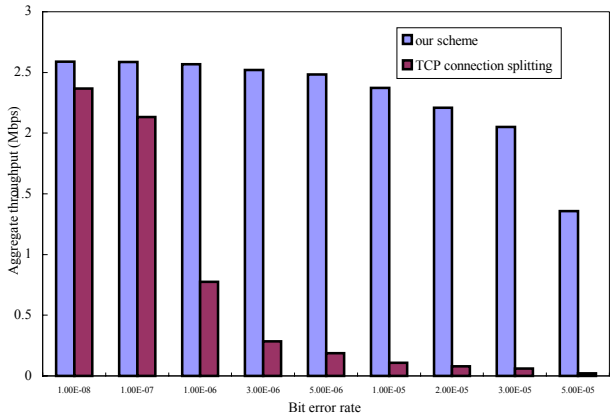

Figure 9 Aggregate throughput for different bit error rates

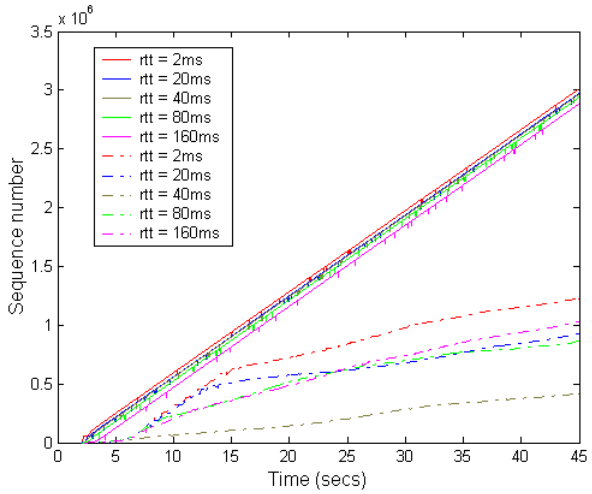

Figure 10 Received sequence number for BER =10^{-6}

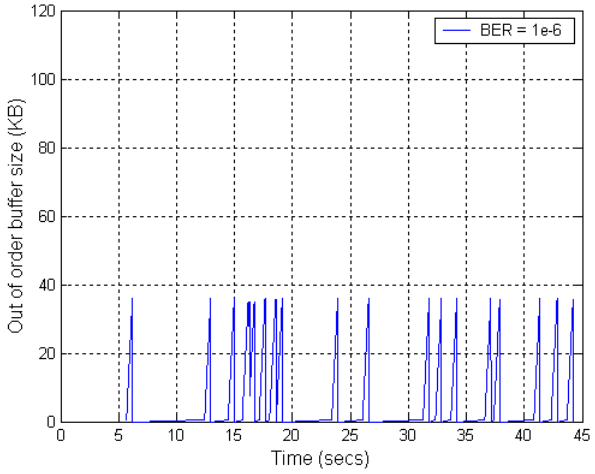

Figure 11 Out of order buffer size for BER = 10^{-6}

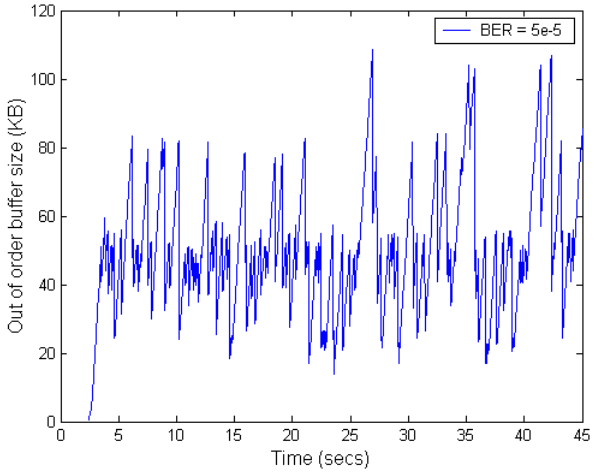

Figure 12 Out of order buffer size for BER = 5e-5

For very low error rate such as 10^{-7} and 10^{-8}, the fairness indexes we computed are very high for both schemes. While for error rate higher than 10^{-5}, the performance is so poor for TCP connection splitting scheme that fairness does not mean too much. Figure 10 plots the received sequence number at the clients for the five satellite connections when BER equals 10^{-6}. The results generated by our schemes are plotted in solid lines while those generated by TCP connection splitting scheme are plotted in dashed lines. It shows that our scheme can improve not only the throughput but also the fairness.

In our scheme, the scheduler continues to send packets as long as there are packets queued at the SGW and there is available buffer at the client. Figure 11 and figure 12 plot the out of order buffer size for the third connection. Figure 11 shows the out of order buffer size for BER equals 10^{-6}. Occasionally there is about one window of packets in the reorder buffer. This means that the error is recovered in one RTT. However when the error rate is too heavy such as 5e-5, retransmissions can get lost again. Figure 12 shows retransmission could be lost twice because sometimes there are about three windows of out of order packets. In order to enable the scheduler to continue sending new packets, the client receiver buffer should be set about four times the bandwidth delay product of the satellite connection.

2) Average response time for short transfers

In addition to the bulk file transfers, another popular application is web browsing, which is characterized by the clients send small requests and the servers reply with small files. We use the same topology as in last section to test the average responsive time for short transfers. By response time, we mean the time interval between the beginning of connection establishment and the time when the last segment of the file is received. For HTTP1.0, each file of the web page requires a separate connection. While for HTTP1.1, a single persistent connection is used for all the files of the web page. For HTTP1.1, the response time we are interested in is that of the first file.All the links are error free so the response

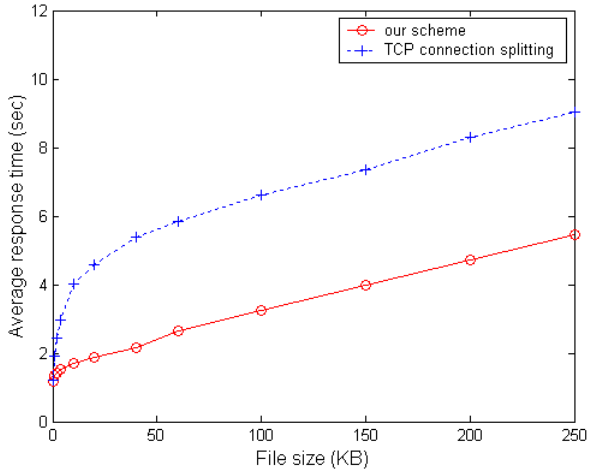

Figure 13 Average responsive time for short transfers

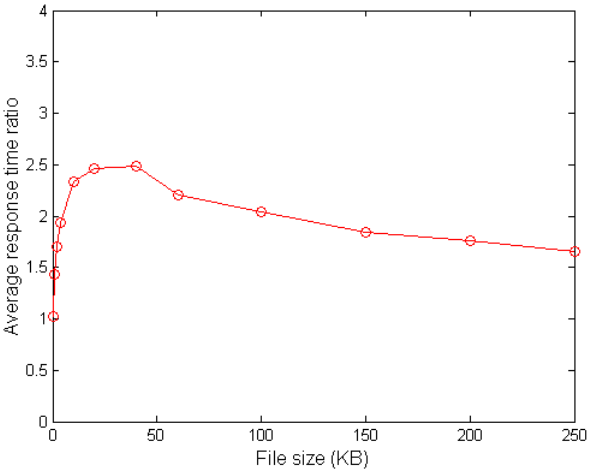

Figure 14 Average response time ratio for short transfers

times are the best case. Figure 13 shows the average response time for the third connection. If the file size is just one segment, both schemes have the same average response time (i.e. ratio=1 point in figure 14). When the file size increases, the slow start phase becomes a dominant portion of the end-to-end response time for TCP connection splitting scheme. While in our scheme, the packets are not limited by the congestion window and can be sent over the satellite link as long as the link is available. For these small files, our scheme can perform 2 to 2.5 times better (figure 13 and figure 14). When the file size is more than 150K bytes, the transfer is more or less like a bulk transfer and the slow start phase is beginning to be amortized by the long transfer time and the response time increases almost linearly for both schemes.

VI. Related work

TCP peach [14] is an end-to-end scheme and it has two new algorithms sudden start and rapid recovery, which replace the slow start and fast recovery algorithm in TCP Reno respectively. Essentially TCP Peach has two logical channels, one is for the data transmission and another one is for bandwidth probing. TCP peach uses low priority dummy segments to probe the bandwidth in sudden start and rapid recovery. The problem with TCP peach is that dummy segments do not carry any information and they are overhead to the data. Another problem is that all the routers need to implement some kind of priority mechanism, which makes it difficult to deploy.

Space communication protocol standards-transport protocol (SCPS-TP) [15] is a set of TCP extensions for space communications. This protocol adopts the timestamps and window scaling options in RFC1323 [7]. It also uses TCP Vegas low-loss congestion avoidance mechanism. SCPS-TP receiver doesn't acknowledge every data packet. Acknowledgements are sent periodically based on the RTT. The traffic demand for the reverse channel is much lighter than in the traditional TCP. However it is difficult to determine the optimal acknowledgement rate and the receiver may not respond properly to congestion in the reverse channel. It does not use acknowledgements to clock out the data rather it uses an open-loop rate control mechanism to meter out data smoothly. SCPS-TP uses selective negative acknowledgement (SNACK) for error recovery. SNACK is a negative acknowledgement and it can specify a large number of holes in a bit-efficient manner.

Satellite transport protocol (STP) [8] adapts an ATM-based protocol for use as a transport protocol in satellite data networks. STP can get comparable performance to TCP SACK in the forward path with significantly less bandwidth requirement in the reverse path. The transmitter sends POLL packets periodically to the receiver, the receiver sends STAT packet as acknowledgements and the reverse path bandwidth requirement depends mainly on the polling period, not on the forward path data transmission rate. Therefore the bandwidth demand for the reverse path decreases dramatically. STP uses a modified version of TCP slow start and congestion avoidance algorithms for its congestion control. While in our scheme we use a round robin scheduler for the congestion control.

Because GEO satellite channel is a FIFO channel, there is no out-of-order routing. And congestion over the satellite link is impossible if the packets are sent at the rate of the satellite bandwidth. In [16], a connection splitting based solution is proposed to use one duplicate ACK to trigger the fast retransmission at the satellite gateway (SGW) and to use a fixed window size for the satellite TCP connection. If there is only one connection in the system, the fixed window can be set to the satellite bandwidth delay product. However multiple connections

with different terrestrial round trip time and different traffic arrival pattern have not been addressed. The paper proposes a new sender algorithm using the same idea as in TCP new Reno [17]. It uses partial ACKs to calculate the bursty loss gap and sends all the potentially lost packets beginning from the partial acknowledgement number. Although it is possible that the sender could retransmit packets that have already been correctly received by the receiver, it is shown that this algorithm performs better than TCP SACK in recovering bursty errors.

VII. Conclusions And Future Work

Because it is difficult for an end-to-end scheme to solve the problems in the satellite hybrid networks, we purpose a connection splitting based solution. A reliable protocol, which decouples the congestion control and error control, is designed for the satellite connections by taking advantage of the specific characteristics of the satellite networks. A SACK based flow control scheme is used to maintain smooth traffic flows. Our results show that our scheme can improve the performance of both bulk and short transfers over the GEO satellites.

Connection splitting does break the end-to-end semantics of TCP. However many applications such as FTP use application layer acknowledgements in addition to the acknowledgements provided by TCP. Therefore the connection splitting based solution still preserves the end-to-end reliability of TCP [18]. Because satellite gateway needs to access the TCP header for connection splitting, it will not work if IPSEC is used. One possible way out is layered IPSEC technique [19]. TCP header in the packet is encrypted with one key, and the data of the packet is encrypted with a different key. The satellite gateway only has the key to decrypt the TCP header. Because the satellite link is still a scarce resource, lossless compression can be used to improve the efficient use of the satellite link. The encryption, compression and the checksum computation for connection splitting are all expensive operations. It has been shown that the processing time without compression and encryption is small and a moderate machine can adequately support numerous split connections with little performance degradation [20]. Future work will address this scalability problem by taking into account all processing overhead.

References

1. V. Arora, N. Suphasindhu, J.S. Baras, D. Dillon, "Asymmetric Internet Access over Satellite-Terrestrial Networks", *CSHCN Technical Report 96-10* available at http://www.isr.umd.edu/CSHCN
2. Partridge and T. Shepard, "TCP performance over satellite links," *IEEE Network*, vol. 11, pp. 44–49, Sept. 1997.
3. M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window," *Internet RFC 2414*, September 1998.
4. M. Allman *et al.*, "Ongoing TCP research related to satellites," *RFC2760*, Feb. 2000.
5. Padhye, J.; Firoiu, V.; Towsley, D.F.; Kurose, J.F. "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transaction on Networking*, April 2000.
6. T. Henderson, E. Sahouria, S. McCanne, and R. Katz, "On improving the fairness of TCP congestion avoidance," in *Proc. IEEE GLOBECOM'98 Conf.*, 1998.
7. V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," *Internet RFC 1323*, 1992.
8. T. R. Henderson and R. H. Katz, "Transport protocols for Internet-compatible satellite networks," *IEEE J. Select. Areas Comm.*, vol. 17, pp.326–344, Feb. 1999.
9. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," *Internet RFC 2018*, 1996.
10. H. Balakrishnan, V. Padmanabhan, and R. Katz, "The effects of asymmetry on TCP performance," in *Proc. 3rd ACM/IEEE MobiCom Conf.*, Sept. 1997, pp. 77–89.
11. Xiaoming Zhou and John S. Baras, " TCP over GEO satellite hybrid networks", in *Proc. IEEE Milcom Conf.* 2002.
12. Kung, H. T. and K. Chang, Receiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks*, Proceedings of INFOCOM '95*, April 2-6, 1995, pp. 239-252.
13. N Samaraweera and G Fairhurst, Reinforcement of TCP/IP Error Recovery for Wireless Communications, *Computer Communications Review (CCR)*, 1998. 28(2), pp30-38.
14. Akyildiz, I.F., Morabito, G., Palazzo, S., "TCP Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 3, June 2001
15. R. C. Durst, G. Miller and E. J. Travis, "TCP extensions for space communications," *Proc. ACM Mobicom*, '96, Nov 1996.
16. I. Minei and R. Cohen "High-speed internet access through unidirectional geostationary satellite channels", *IEEE J. Select. Areas Commun.*, Vol. 17 Feb 1999.
17. S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *Internet RFC 2582 (Experimental),* April 1999.
18. A. Bakre and B. R. Badrinath, "Implementation and performance evaluation of indirect TCP" *IEEE Transactions on Computers* Vol. 46 No. 3, March 1997.
19. Y. Zhang, B. Singh, "A Multi-Layer IPsec Protocol," Proc.proceedings of 9th USENIX Security Symposium, Denver,Colorado, August 2000. http://www.wins.hrl.com /people/ygz/papers/usenix00.html
20. K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Comput. Commun. Rev.*, vol. 27, pp. 19–43, Oct. 1997.