TCP SPLITTING PROTOCOL FOR BROADBAND AERONAUTICAL SATELLITE NETWORK

Yadong Shang, Michael Hadjitheodosiou, Center for Satellite & Hybrid Communication Networks, Institute for Systems Research, University of Maryland, College Park, MD 20742

Abstract

In this paper, we recommend suitable transport protocols for an aeronautical network supporting Internet and data services via satellite. For the future IP-based aeronautical satellite hybrid network, we describe TCP protocol and focus on the problems that cause dramatically degraded performance of the Transport Protocol. Based on the observation that it is difficult for an end-to-end solution to solve these problems effectively, we propose a new TCP-splitting protocol, termed Aeronautical Transport Control Protocol (AeroTCP). The main idea of this protocol is to use a fixed window for flow control and one duplicated acknowledgement (ACK) for fast recovery. We also study the Gilbert-Elliott channel model for satellite channel since protocol behavior often depends strongly on the channel behavior. Our simulation results show that AeroTCP can maintain higher utilization for the satellite link than end-to-end TCP, especially in burst error environment.

Introduction

America's aviation industry is soaring into the 21st Century with projected increases in business, recreation, and personal travel. According to the Federal Aviation Administration (FAA) *Aerospace Forecasts 2002-2013*, domestic operations (e.g., takeoffs and landings) are expected to increase from over 66 million in 2001 to more than 81 million in 2013 [1].

The National Airspace System (NAS) is one of the safest and most secure aviation systems in the world. However, modernization is required for several reasons. First, the move toward Free Flight concepts requires equipment with new capabilities and new procedures. Secondly, many current systems are aging. These systems, which have provided years of reliable service, must be maintained to ensure safe operations while also transformed to future systems. Finally, NAS modernization programs leverage technological advance to improve Communications, Navigation, and Surveillance (CNS) and automation systems to support projected air traffic growth.

Satellite network is recognized as one of the technologies to meet the needs of future aeronautical communications for its significant improvements in over-ocean coverage. The expected advantages of the satellite systems for aeronautical communications also include high communication capacity, low message propagation delay, suitability to free flight concepts, and other economic benefits [2].

Several companies (e.g., Inmarsat, Iridium, Boeing) have announced plans to use satellite technologies to provide commercial broadband data services for airline passengers [3, 4]. The future aeronautical satellite systems will offer Internet connections at up to broadband (tens of Mbps) data rates via networks of GEO or LEO satellites. Figure 1 illustrates the general topology of aeronautical satellite network. This system will be composed of three major segments: cabin segment with on-board networks, space segment for interconnection of the cabin with the terrestrial networks, ground segment which provides the interconnection to the terrestrial personal and data networks as well as the Internet backbone.

These systems are expected to offer Internet access and to support virtual private networks (VPN) for passengers in flight. However, the performance of data communications protocols and applications over such systems is the subject of heated debate in the research community, especially the transport protocol in the Internet TCP/IP protocol suite [5]. Some researchers insist that TCP will work suitably in a satellite environment, while others have suggested satellite specific protocol options for improved performance, and still others claim that TCP cannot work effectively over satellite channels.

0-7803-8539-X/04/$20.00 © 2004 IEEE

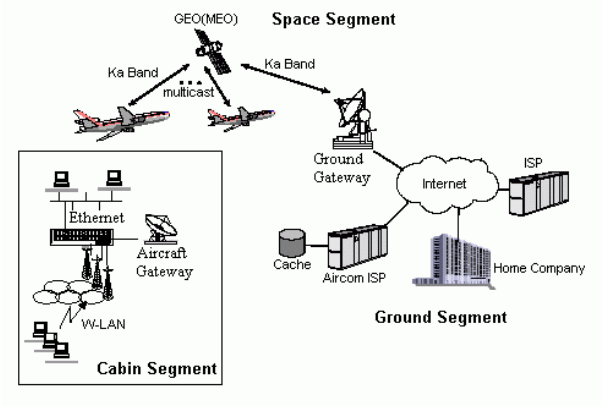

Figure 1. Aeronautical Satellite Network

In this paper, we will design a new splitting based TCP protocol for aeronautical satellite network. In section 2, we will first describe the standard TCP operations. In section 3, The TCP problems in satellite networks are discussed in details. We also list some proposed solutions and their limitations. Based on the observation that the end-to-end TCP solution cannot solve those problems effectively, we proposed a new splitting based TCP protocol in section 4. In section 5, we discuss the channel model used in our simulation because the performance of protocols often depends on channel behavior. Our simulation results in section 6 shown that our splitting protocol can maintain high utilization of satellite channel and has better performance than end-to-end solutions. Section 7 concludes the paper.

TCP Protocol

TCP is a connection-oriented, end-to-end, process-to-process reliable transport protocol [6]. There are several mechanisms in TCP to ensure those functions, which are flow control, congestion control and error control.

Flow control: TCP uses a sliding window to achieve flow control. The TCP receiver sets the receive window (RCVWND) field in the acknowledgement to its free buffer size so that the sender will never overflow the receiver's buffer.

Congestion control [7]: The TCP sender maintains a state variable CWND for congestion window size. While RCVWND is used to guard that the sender will not overload the receiver buffer, the CWND is used to guard that the sender will not overload the network. The TCP sender can send at most the minimum of RCVWND and CWND window worth packets without receiving any ACK. In the most popular TCP Reno, there are four algorithms used for congestion control, which are slow start, congestion avoidance, fast retransmit, and fast recovery. Slow start is used upon the start of a new connection to probe the network bandwidth, it increases the CWND by one Maximum Segment Size (MSS) when an ACK is received, which results in increasing congestion window exponentially. TCP stays in slow start until its CWND is greater than the slow start threshold. After that TCP gets into congestion avoidance, it increases CWND about one MSS per round trip time (RTT). Fast retransmit algorithm is triggered when a fixed number of duplicate acknowledgements (usually 3) are received. TCP retransmits the potential lost packet indicated by the acknowledgement and cuts its CWND to half. After that, it inflates its CWND by one MSS when a duplicate acknowledgement is received. If there is one and only one packet lost in a single window, the inflation can increase the CWND to the original CWND before the loss after about half RTT. After that TCP can send a new packet when each duplicate acknowledgement is received if allowed by the RCVWND. Finally it will send half a window new packets when it receives the first non-duplicate acknowledgement.

Error control: Error control is the main component of reliable protocols, which includes error detection and error recovery. TCP uses acknowledgement packet, timer and retransmission to achieve error control. TCP uses cumulative acknowledgement, which means when a packet gets lost, it prevents the acknowledgement from being advanced and the window cannot slide until the lost packet is recovered. The sliding window mechanism actually ties the flow control, congestion control and error control together and it becomes vulnerable when there are losses due to congestion loss and packet corruptions in the network.

TCP Problems in Satellite Network

The main characteristics of the aeronautical satellite channel that affect transport protocol performance are long propagation delay, large

bandwidth delay product, high bit error rate, and bandwidth asymmetry. If part of the communication path includes a satellite channel, these parameters can vary substantially from those found on wired networks.

Long Propagation Delay

GEO satellite is about 36,000km above the earth. The propagation delay from the earth up to the satellite and from the satellite down to the earth is about 125ms. Therefore a typical round trip time (RTT) for two-way system is about 500ms plus the delay for terrestrial networks.

For a connection with such large RTT, TCP spends a long time in slow start before reaching the available bandwidth. The time taken by TCP slow start to reach the satellite bandwidth (SatBW) is about $RTT \times \log_2(SatBW \times RTT)$ when every TCP segment is acknowledged. For short transfers, they could be finished in slow start, which obviously does not use the bandwidth efficiently. Some researchers propose to use a *large initial window* [8] up to 4380 bytes (or a maximum of 4 segments) rather than 1 segment. Files less than 4K bytes (many web pages are less than this size) can finish their transfers in one RTT rather than 2, 3.

Large Bandwidth-Delay Product

For GEO satellite operating on K/Ka band, the bandwidth delay product (e.g., 20Mbps*580ms RTT) in this system is very large comparing to that in terrestrial network. To fully utilize such channel, we need to put that much of data (equal to bandwidth-delay product) into the link. TCP will send new data until it receives the ACKs for old data. That means TCP window should be at least the bandwidth delay product.

In TCP protocol syntax, the receiver advertised window in the TCP header cannot be more than 64K bytes, which limits the two-way throughput to roughly 1Mbps in GEO satellite networks. *Window Scaling* [9] is proposed to deal with this problem, which significantly increases the amount of data that can be outstanding on a connection by introducing a scaling factor to be applied to the window field.

However, Window Scaling does not solve all problems. 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.

High Bit Error Rate

In satellite channel, bit error rate of the order of 10^{-6} are often observed. This is primarily because the existing systems with legacy equipment and many existing transponders were optimized for analog voice and video services. New modulation and coding techniques, along with higher-powered satellites, should help to make bit error rate very low for GEO system.

Because TCP Reno (popularly used in many systems) treats all losses as congestion in the network, the link layer error can causes TCP to drop its window to a small size and leads to poor performance. TCP SACK can convey non-contiguous segments received by the receiver in the acknowledgements so that the sender can recover error much faster than TCP Reno, which well know can recover only on loss per RTT.

Bandwidth Asymmetry

Satellite networks can be asymmetric in several ways. Some satellite networks are inherently bandwidth asymmetric, such as those based on a direct broadcast satellite (DBS) downlink and a return via a dial-up modem line. For purely GEO or LEO system, bandwidth asymmetries may exist for many users due to economic factors.

With respect to transport protocols, the forward throughput achievable depends not only on the link characteristics and traffic levels in the forward path but also on those of the reverse path. The congestion in reverse path could lead to poor performance in the forward link because TCP uses ACKs to clock out data. To alleviate this problem, ACK filtering was proposed to drop the ACKs in the front of the IP queue by taking advantage of the cumulative acknowledgement strategy in TCP [10]. The situation is even worse for two-way transfers. When the users are sending data 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.

From above description, we can see that TCP did not perform well over satellite networks (or high latency networks in general) for a number of reasons related to the protocol syntax and semantics. Over the past decade, a number of TCP flavors and extensions have been specified which improve the performance of the TCP protocol in satellite environments [9,11]. Most of those flavors and extensions are end-to-end enhancements to standard TCP and they cannot solve all problems, or not very effectively [12]. 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 aircraft platform. Considering the interoperability issue, we propose a connection splitting based scheme to solve those problems.

Splitting Protocol

The idea behind split connections is to shield high-latency or noisy network segments from the rest of the network, in a manner transparent to applications. Figure 2 illustrates the general split case, in which an end-to-end TCP connection is split into 3 connections at the aircraft gateway and ground gateway. One connection is from the Internet server to the ground gateway, another one is from the ground gateway to the aircraft gateway, and the last one is from the aircraft gateway to the client in aircraft.

We consider the data transfer from the Internet servers to the client in aircraft. Ground gateway sends premature acknowledgements to the Internet servers and takes responsibility to relay all the acknowledged packets to the aircraft gateway reliably. The aircraft gateway does the same job to relay the data to the client. For the satellite link between the ground gateway and the aircraft gateway, a satellite optimized transport protocol can be used.

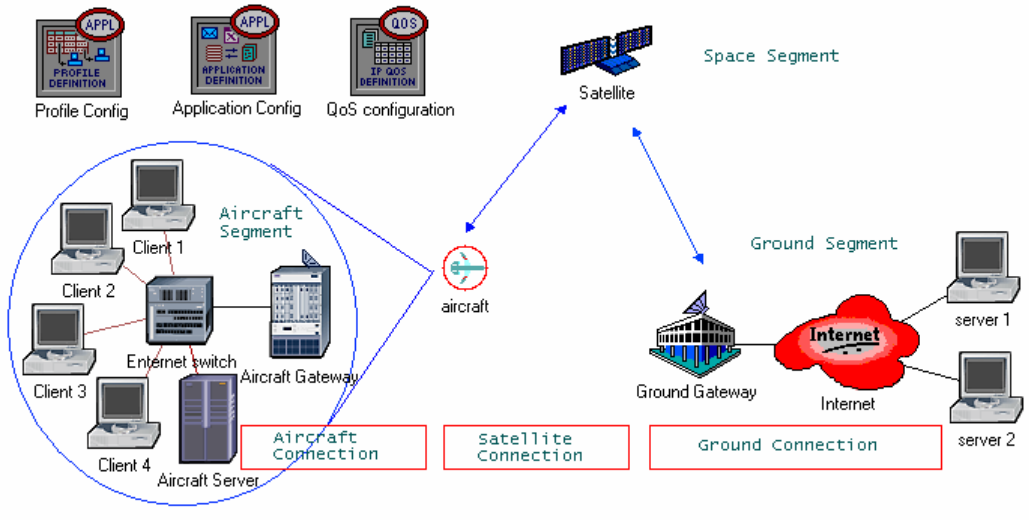

Figure 2. TCP Splitting Protocol for Aeronautical Satellite Networks

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. 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. We design a new TCP splitting protocol, which we called Aeronautical Transport Control Protocol (AeroTCP), for the satellite connection. The main idea is to use one duplicate ACK to trigger the fast retransmission at the satellite gateway and to use a fixed window size for the satellite TCP connection. This implementation of this idea will be discussed in detail in the following.

Flow Control: We still use a sliding window for flow control, however, here the window are fixed for each satellite connection. For a normal router, they only have the functions up to IP layer, while the satellite gateway is able to process TCP packets. All the TCP packets received from the servers are forwarded to the TCP received buffer of the ground gateway and they are moved from the received buffer to send buffer for transmission. The receive buffer and send buffer can be implemented by one physical buffer.

The buffer size assigned to each connection at the satellite gateway has a direct impact on the end-to-end TCP throughput. Consider the traffic from Internet server to the client on aircraft, assume there is only one connection in this system, the buffer size assigned to the TCP connection is *Buff* and the effective satellite bandwidth is *SatBW*. The date in the satellite pipe is *SatWin* and the advertised receiver window for the server is *RecvWin*. The round trip time for the satellite connection is *SatRTT* and for the ground connection is *GndRTT*. Then the system reaches the steady state, the input rate of the queue at the ground gateway should be equal to the output rate of the queue, i.e., *RecvWin / GndRTT = SatWin / SatRTT*. The throughput of the connection is *min (SatBW, Buff / (SatRTT + GndRTT))* and the backlog packets are *max (o, Buff – SatBW * (SatRTT + TerrRTT))* [13]. From the above analysis, 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. For simplicity, we 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 that peak rate. We assume large but not infinite buffer is available at the client and the TCP flow control is still enforced so that the gateway will not overflow the receiver's buffer.

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. The gateway can send packets as long as there are packets in the buffer and the receiver's window allows sending. Also there is no need to exponentially back off the timer after timeout because congestion is impossible over the satellite link. Timer is used only for error recovery. As long as there are packets buffered at the satellite gateway, the satellite link can be fully utilized. 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. This way satellite link efficiency is achieved.

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. 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 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 for error control by using one duplicated ACK for fast recovery. We keep track of the packets in sequence space of all acknowledged packets. Whenever a duplicated acknowledgement is received, we just assume that packet is lost and retransmit it. During the recovery, we use the same idea as in TCP NewReno [14]. We use partial ACKs to calculate the burst loss gap and send 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 more effective in recovering burst errors (popular in satellite environment). Timer is still used as the last resort for loss recovery, however, after timer expires, two copies of the lost packet are sent to increase redundancy.

Gilbert-Elliot Channel Model

For simulation of communication protocols, channel modeling is an important task since the protocol behavior often depends strongly on the channel behavior. In this paper, we consider satellite radio transmission in K/Ka band. It is widely accepted that this radio channel is an error prone channel with non-stationary error characteristics. Bit error rates as bad as $10^{-2\sim-3}$ are reported. The error process in general is constituted by different phenomena:

- Path Loss
- Fast Fading due to movement and multi-path propagation.
- Slow Fading due to moving beyond large obstacles.
- Noise and Interference from other networks or devices like microwave ovens.

For modeling the error characteristics of a wireless channel, a simple and widely used model is the *Gilbert-Elliot model* [15]: Consider a two state Markov chain with the states named *Good* and *Bad* as shown in Figure 3. Each state represents a different binary symmetric channel (BSC) with a specific constant bit error rate (BER), e_G in the good state, e_B in the bad state ($e_G << e_B$). Within one state, bit errors are assumed to occur independently from each other. The bit error rate in general depends on the frequency and coding scheme used and on environmental conditions (e.g., number of paths between source and destination).

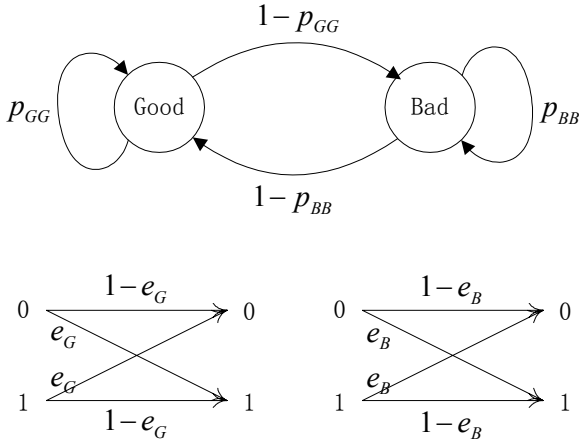

Figure 3. The Gilbert-Elliot Model

In the literature, a discrete time Markov chain is often used, with state transitions after every channel symbol. For complete specification of the discrete model, the two bit error probabilities and the 2×2 stochastic state transition matrix are sufficient. The state transition is completely determined by the values p_{GG} (for the probability that the next state is the good state, given that the current state is also the good state) and p_{BB}. In

practice, those parameters are chosen such that the error statistics of channel can be represented fairly accurate. The error gap distribution and the block error statistics are used to characterize the error bursts of channels with memory [16].

Assume we have the error sequence generated from Gilbert-Elliot model with 1 means error and 0 means no error. An error gap is defined as a string of consecutive zeros between two ones in the error sequence. The error gap probability is the probability that there are at least *n* error free bits between errors. The block error probability $P(m,n)$ is the probability that *m* errors occur in a block of *n* consecutive bits.

Using the results of the ACTS propagation experiments, published in [17], for a fade attenuation threshold of 10dB, we have $p_{GG} = 0.9999813$, $1 - p_{BB} = 0.00172$. To calculate the bit error probability at the output of the channel decoder, we use the link budget calculations of a commercial satellite system proposed in [18]. The calculations confirm that for signal attenuation of less than 10 dB, link budget and channel coding are capable of keeping the bit error probability around 10^{-9}. For signal attenuation of more than 10dB, bit errors occur with probability 0.1. Thus for our simulation, we choose $e_G = 0$, $e_B = 0.1$.

A bit error simulation model based on the Gilbert-Elliot model can be quite easily implemented as following:

1. Perform a random experiment according to the present channel state to determine whether the bit is erroneous. If the bit is erroneous, mark that bit.
2. Perform a random experiment according to the present channel state to determine the next state. In case a state change occurs, set the state variable and error probability to the new state.
3. Wait one bit time and continue with step 1.

The drawback of this modeling approach is that at every bit time, two Bernoulli experiments have to be executed: one to determine whether a bit error occurs and the other one to determine a state change. This will slow down packet level communication protocol simulation, since a packet can consist of thousands of its.

Two methods are used to speed up the bit error simulation [19]. The first is to attack the calculation of the state change by means of a Bernoulli experiment at every bit time. Applying some statistics leads to an improvement. For instance, one can count the number of independent Bernoulli experiments until a success event occurs. This number follows a geometric distribution. We can use this knowledge to speed up the simulation. Instead of executing every bit time a Bernoulli experiment to check a state transition in the Markov chain, we derive from the geometric distribution the number of time steps of next state transition.

Next, we attack the calculation of bit errors. Every bit is evaluated with a Bernoulli experiment. Consider the case where one only wants to know whether the packet is correct or wrong, we can use the packet error probability in a single Bernoulli experiment to decide about correctness of the packet. If one knows the bit error probability and the packet length in bits, it is possible to compute the packet error probability. Of course the channel state may change during a packet transmission. This can be handled by computing the error probability only of the packet fraction, which belongs to the current channel state, and for the remaining packet fractions repeat this calculation. The partial results can be combined to determine the packet error probability.

Simulation Results

To test our splitting scheme, we setup a simulation model as shown in figure 4. A client downloads a file of 1.6M bytes from a server via a satellite link. The link delay between the hybrid gateway and the client is 250ms and the link delay between server and gateway is 40ms. Therefore the RTT between the server and the client is 580ms. The satellite link bandwidth is DS1 (1.544Mbps) and the ground link is 10Mbps. The transport protocol for the ground connection is normal TCP SACK, while we use our TCP splitting scheme for the satellite connection. We assume the satellite link between the gateway and the client is noise channel with Gilbert-Elliot model, while the ground link has

no error. The statistics we collect is link utilization and link throughput of the two downstream links.

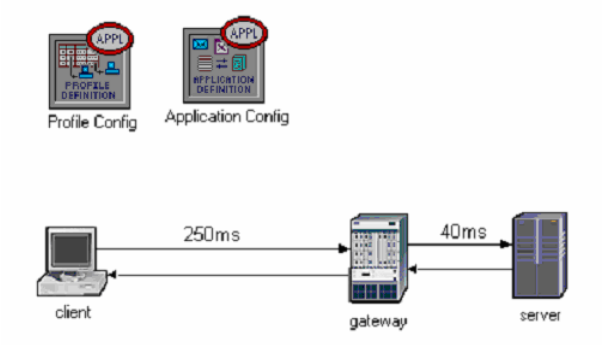

Figure 4. Experiment Setup for TCP Splitting

Figure 5 shows the utilization 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 since the TCP can actually operate with large window. For TCP connection splitting scheme, when the bit error rate increases up to 10^{-6}, the link layer corruption causes the satellite TCP to drop its congestion window, which leads to degraded performance. When the BER increases to 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 and the link utilization is very low. While for our scheme, the TCP can send packet at the rate of bandwidth as long as there are packets in the buffer and the receiver has enough buffer. The utilization is drop when the BER increases to 10^{-5}, which is because lots of packets are lost due to layer error corruptions.

It is important to compare our scheme with the end-to-end TCP solutions. Figure 6 shows the FTP response time for end-to-end TCP, TCP splitting, and our scheme to download a file of 1.6M bytes. The TCP splitting uses TCP SACK for both the satellite connections and terrestrial connections. We can see that the TCP splitting has better performance than end-to-end TCP because in TCP splitting, the terrestrial connection can operate with large window and send packets to gateway faster due to no error. It is interesting that performance is improved if we just use standard splitting protocol, although not noticeable when the BER is high. In the other hand, our scheme has the best performance than both end-to-end TCP and TCP splitting. This is because both the satellite connection and terrestrial connection of our scheme can operate with large window. The response time is more obvious when the BER increase to 10^{-5}.

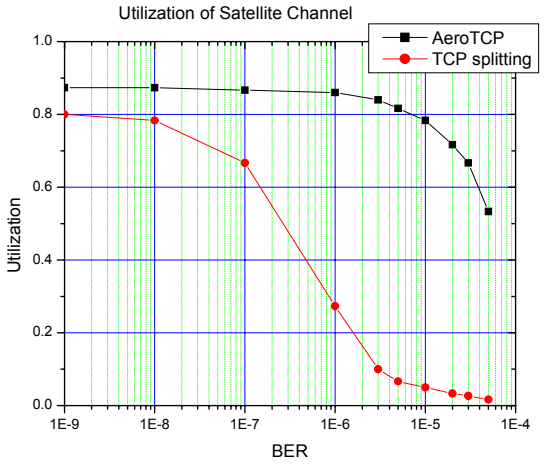

Figure 5. Utilization for Different Bit Error Rates

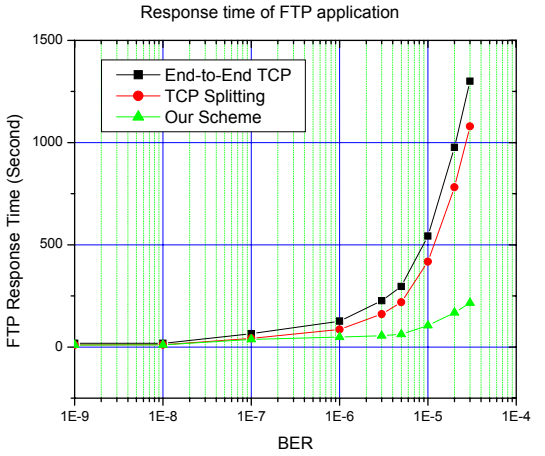

Figure 6. Response Time for FTP Application

Conclusions

In this paper, we have investigated the performance of IP-Compatible transport protocols over satellite links from several perspectives. We observed degradation in TCP performance for large bandwidth-delay product networks such as aeronautical satellite systems. We describe the TCP

problems in satellite networks and its possible solutions. Because it is difficult for an end-to-end TCP solution to solve the problems in the aeronautical satellite networks, we propose a connection splitting based solution, AeroTCP, which is designed for the satellite connections by taking advantage of the specific characteristics of the satellite networks. Our simulation results show that our scheme can maintain high utilization of the satellite link and has better performance than end-to-end solutions.

Acknowledgements

This work is supported by the Center for Satellite and Hybrid Communication Networks, under NASA cooperative agreement NCC8-235 and NASA cooperative agreement NAG3-2844.

References

[1] Blueprint for NAS Modernization 2002 Update, http://www.faa.gov/nasarchitecture/Blueprint2002.htm

[2] Oagur Ercetin, Michael O. Ball, Leandros Tassiulas, "Next Generation Satellite systems for Aeronautical Communications", Technical Research Report of National Center of Excellence in Aviation Operations Research, NEXTOR T.R. 2000-1, ISR T.R. 2000-20

[3] Peter W. Lemme, Simon M. Glenister, Alan W. Miller, "Iridium Aeronautical Satellite Communications", IEEE AES System magazine, November 1999

[4] W. H. Jones, M. de La Chapelle, "Connexion by BoeingSM-Broadband Satellite Communication System for Mobile Platforms", MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE, Volume: 2, 2001 Page(s): 755 -758 vol.2

[5] W. Stevens, "TCP/IP Illustrated", Volume 3, Addison Wesley, 1996

[6] J. Postel, "Transmission Control Protocol", Internet RFC 793, 1981.

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

[8] M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window", Internet RFC 2414, September 1998.

[9] M. Allman, D. Glover, and l. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", RFC 2488, 1999

[10] H. Balakrishnan, V. Padmanabhan, and R. Katx, "The Effects of Asymmetry on TCP performance", Proceedings of Third ACM/IEEE MobiCom Conference, pp. 77-89, Sept. 1997.

[11] M. Allman(ed), S. Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke, "Ongoing TCP research Related to Satellites", RFC 2760, 2000

[12] Yadong Shang, Michael, Hadjitheodosiou, and John Baras, "Using Broadband Satellite Systems to Support Aeronautical Communications", Proc. 22nd International Communications Satellite Systems Conference and Exhibit, Monterey, California, 9-12 May 2004.

[13] Xiaoming Zhou and John S. Baras, "TCP over GEO satellite hybrid networks", in Proc. IEEE MilCom Conference, 2002.

[14] S. Floyd and T. Henderson, "The New Reno Modification to TCP's Fast Recovery Algorithm", Internet RFC 2582 (Experimental), April 1999.

[15] E. O. Elliott, "Estimates of Error Rates for Codes on Burst-Noise Channels", Bell Syst. Tech. J., Vol. 42, pp. 1977-1997, Sept. 1963.

[16] Jeffrey Slack, "Finite State Markov Models for Error Bursts on the Land Mobile Satellite Channel", Master Thesis, Brigham Young University, August 1996.

[17] J. Slack, M. Rice, "Finite State Markov Models for Error Bursts on the ACTS Land Mobile Satellite Channel", NASA ACTS Experiment paper ACTS-96-046, 1996

[18] E. J. Fitzpatrick, "SPACEWAY System Summary", Space Communications, Vol. 13, pp. – 23, 1995

[19] Jean-Pierre Ebert, Andreas Willing, "A Gilbert-Elliot Bit Error Model and the Efficient Use in Packet Level Simulation", TKN Technical Reports Series, March 1999