AeroTCP: A Splitting Transport Protocol for an IP-based Satellite Network Supporting Aeronautical Communications

Yadong Shang*, Michael Hadjitheodosiou† and John Baras‡
Center for Satellite & Hybrid Communication Networks, Institute for Systems Research, University of Maryland, College Park, MD 20742, USA

The IP-based Broadband Aeronautical Satellite Network will provide numerous new applications and services for both airspace system operations and passenger communications. However, the performance of data communications protocols and applications over such systems is dramatically degraded, especially the Internet TCP/IP protocol suite. In this paper, we propose a new transport protocols for satellite network supporting Internet and data services. Based on the observation that it is difficult for an end-to-end TCP solution to solve the performance problem effectively, we propose a new splitting based TCP protocol, called Aeronautical Transport Control Protocol (AeroTCP), which takes advantage of the specific properties of satellite channel. Our simulation results show that AeroTCP improves satellite channel utilization and fairness.

I. Introduction

World's aviation industry is soaring into the 21st Century with projected increases in business, recreation, and personal travel. The current airspace systems are quickly becoming overburdened by increases in air traffic coupled with the use of old technologies and legacy systems [1]. These systems must be maintained to ensure safety and efficiency while also transitioning to future systems. In addition, airplanes seem to be the last remaining islands where mobile communications and Internet access is not available. The demand for making air travel more pleasant, secure and productive for passengers also demonstrated the need for major improvements and new initiatives in aeronautical communications.

Inspired by the big market and business opportunity, many investigations and commercial activities are being developed to establish broadband aeronautical communication networks. 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 natural broadcast capability, high communication capacity, suitability to free flight concepts or for mobile users, and other economic benefits [2].

Several companies (e.g., Inmarsat, 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. In this paper, we focus on the GEO satellites because of its large covering area and significant reduction in system complexity compared to LEO satellite system. Figure 1 illustrates the IP-based network 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. In our paper, the GEO satellites are bent pipe satellites, which are simply signal repeaters in the sky. They are physical layer devices and no switching is performed on board.

---

* Research Assistant, CSHCN, University of Maryland.
† Assistant Research Scientist, CSHCN, University of Maryland, AIAA Member.
‡ Professor, CSHCN, University of Maryland, AIAA Member.

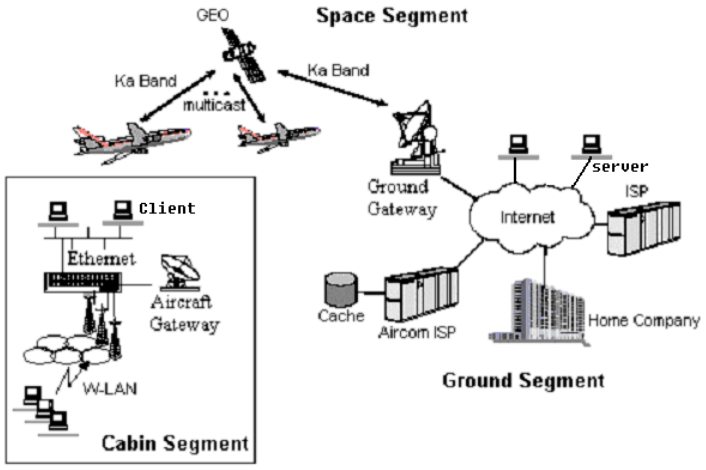

Figure 1 Aeronautical Satellite Network

The aeronautical satellite networks will provide numerous new applications for both airspace system operations and secure air traffic management. Those applications could include the System Wide Information Management (SWIM), Controller Pilot Data Link (CPLDC), regular downloading of the aircraft's "black box" data, better enhanced weather information, voice over IP, telemedicine, and electronic flight bag applications. It also provides airlines with new revenue generating services (e.g. entertainment services, Internet access, directed advertising, and telephone service) for passengers.

Application of commercial off the shelf technologies and techniques has the potential to make network operations economically and technically realizable. However, the performance of data communications protocols and applications over such systems is the subject of heated debate in the research community, especially the Internet TCP/IP protocol suite [5,6]. In this paper, our goal is to design an efficient and fair transport layer protocol. The rest of the paper is organized as follows. In section II, we present the basic background of TCP and its problems in satellite networks. Then we discuss some proposed solutions and related works in section III. The AeroTCP and its implementation are described in section IV. The final section provides simulation results, together with the conclusion and future work.

II. TCP operations and its problems in satellite networks

Current TCP implementations, TCP-Reno, contain a number of algorithms aimed at controlling network congestion. These algorithms include slow start, congestion avoidance, fast retransmit and fast recovery [7]. Together they define the congestion window, *cwnd*, as an estimate of the maximum number of packets that can be sent without receiving any acknowledgement (ACK). While the receiver's advertised window, *rwnd*, is used to guard that the sender will not overflow the receiver buffer, the *cwnd* is used to guard that the sender will not overload the network. The TCP sender never sends more than the minimum of *cwnd* and *rwnd* window worth packets without receiving any acknowledgement.

The TCP sender operates in one of two modes: slow start or congestion avoidance. During slow start the sender starts with a congestion window of one packet and grows it by one with every acknowledgement received. Assuming an acknowledgement is sent for every data packet received, which is not the case when the receiver uses delayed ACKs, this results in doubling *cwnd* every round trip and increasing *cwnd* exponentially. While in congestion avoidance mode, the sender increases the value of *cwnd* by 1/cwnd for every data packet received, which is approximately equivalent to an increase of one packet every round trip time (RTT), yielding linear growth.

When a packet is lost, it triggers a duplicate ACK (dupACK), which carries the same sequence number as a previous acknowledgement. When the third dupACK is received, the sender assumes a packet has been lost and enters fast retransmit mode. TCP transmits the potential lost packet indicated by the ACK and cuts its *cwnd* to half. After that it inflates its *cwnd* by one packet when a dupACK 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 dupACK is received if allowed by *rwnd*. Finally it will send half a window new packets when it receives the first non-duplicate ACK. After receiving an ACK for the retransmitted packet, the sender performs a procedure called fast recovery by shrinking *cwnd* to half and entering congestion avoidance mode.

The fast retransmit mechanism is not always triggered when a packet is lost. As a simple example, consider the case where the window size is only 4, in which case only two dupACKs can be received. To detect a packet loss even in this case, the sender maintains a retransmission timer. When the timer expires, the oldest packet for which

an ACK has not been received is retransmitted. The time the sender is idle, waiting for the timer to expire, is called a timeout. After a timeout, the sender retransmits the lost packet, shrinks *cwnd* to 1 and enters slow start. Thus timeouts have a significant impact on throughput both because they introduce a period of idle time and because they shrink the window.

Aeronautical satellite networks have several characteristics that differ from terrestrial networks. Two main characteristics may degrade the performance of TCP: long round trip delay and burst losses.

The first problem is long round trip delay. A Geosynchronous Earth Orbit (GEO) satellite is about 36,000km above the earth. At this altitude the orbit period is the same as the earth's rotation period. Therefore, each ground station is always able to "see" the orbiting satellite at the same position in the sky. The propagation time for a radio signal to travel twice that distance is about 240ms (corresponding to a ground station directly below the satellite). Therefore, the propagation delay for a round trip time (for a message and the corresponding reply) is about 560ms, including 80ms RTT for typical terrestrial Internet delay. The RTT will be increased by other factors in the network, such as the transmission time and propagation time of other links in the network path and queuing delay in gateways.

Large RTT will result in long slow start and low bandwidth utilization. The time taken by TCP slow start to reach the satellite bandwidth, SatBW, is about $t_{SlowStart} = RTT * (1 + \log_2(SatBW * RTT / l))$, where RTT is the round-trip time and l is the average packet length expressed in bits [8]. This equation is satisfied when every TCP segment is acknowledged. 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.

Large RTT also introduces large delay bandwidth product (DBP) for satellite link. The DBP defines the amount of data a protocol should have "in flight" (data that has been transmitted, but not yet acknowledged) at any time to fully utilize the available channel capacity. However, the receiver advertised window, *rwnd*, is 16 bits in the TCP header. This window cannot be more than 64K bytes, which limits the two-way system throughput to 64KB/560ms, 117KBps. Window Scaling [9] 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 corruption. The multiple losses will trigger TCP congestion control algorithms and lead TCP to actually operate with a small average window.

Large RTT also lead to large timeout value because the timeout value is calculated dynamically, based on the round trip time measurements the sender performs throughout its operation [5]. If a loss can not detected by fast retransmit, the sender is idle waiting for the timer to expire, and operates below its optimum speed a few round trip times afterwards during slow start mode. In this mode, the sender works with a window smaller than 4 for 2 round trip times, even a single packet loss may cause a timeout.

The second problem is burst losses. Communication over satellite links is often characterized by sporadic high bit error rates and burst losses. This is especially true when working in the Ka band (30/20 GHz), where weather conditions greatly affect link availability.

TCP uses all packet drops as signals of network congestion and reduces its window size in an attempt to alleviate the congestion. In the absence of knowledge about why a packet was dropped (congestion or corruption), TCP must assume the drop was due to network congestion to avoid congestion collapse. Therefore, packets dropped due to corruption cause TCP to reduce the size of its sliding window, even though these packet drops do not signal congestion in the network. After any retransmission, whether following a timeout or following fast transmit, the sender shrinks its transmission window to one or to half its original size, respectively. Thus following a loss the sender operates below its optimum speed for a few round trip times. If losses occur at the time the window is growing back towards its optimal size, they lower the window yet again. Moreover if a loss occurs while the window grows in slow start, the growth rate turns from exponential to linear, and it takes even longer for the window to reach the optimal value.

Burst losses are more likely lead timeouts. The probability of a timeout increases with the packet loss rate because for high loss rates, the probability of losing several packets in the same window, which usually leads to

timeouts, increases. TCP does not perform well under burst losses. The mechanism for efficient recovery of lost packets, e.g. fast retransmit, fails when several consecutive packets are lost, drastically affecting the throughput. For TCP Reno, in order for the fast retransmit algorithm to recover the loss, the congestion window size has to be greater than four for single packet loss and has to be greater than ten for two consecutive losses in one window. While for three or more consecutive losses in one window, the TCP sender has to wait for timeout to recover the loss [10]. TCP New-Reno [11] can avoid many of the retransmit timeouts of Reno when a large number of packets are dropped from a window of data. However, New-Reno can only recover one lost packet during each RTT. TCP SACK [12] can convey information about non-contiguous segments received by the receiver in the acknowledgements so that the sender can recover error much faster than TCP Reno and New-Reno.

III. Related work of TCP over satellite network

The proposed TCP solutions for satellite environment can be categorized into four classes: End-to-end enhancements, TCP connection splitting, Rate based solution, and link layer solution. We will discuss some of the proposed solutions related to our work, specially the TCP splitting solutions.

TCP enhancements TCP enhancements include large initial window [13], delayed ACKs after slow start [14], TCP for transaction [15], selective acknowledgement [12], and forward acknowledgement [16]. All these enhancements are end-to-end solutions. They only need to be implemented at the end nodes, rather than at every router in the network. However, it is difficult for an end-to-end solution to solve these problems in the hybrid satellite networks effectively.

TCP Spoofing [17]. A router near the source sends back ACKs for TCP segments in order to give the source the illusion of a short delay path. TCP spoofing improves throughput but has some problems. The router must do a considerable amount of work because it becomes responsible for the correct delivery of the TCP segments it acknowledges to the source. Spoofing requires ACKs to flow through the same path as data. On contrary, in Internet it is very common that ACKs flow through a different path than data. If the path changes or the router crashes, data may get lost. If IP encryption is used, this scheme cannot be applied.

I-TCP [18]. I-TCP stands for Indirect TCP and is mainly designed for mobile Network. The basic idea of indirect TCP is that the end-to-end TCP connection is now divided into two connections, one is from the server to the base station and another one is from the base station to the mobile users. The base station sends premature acknowledgements to the server and takes responsibility to relay the data to the mobile host reliably. The advantages are the separation of flow control and congestion control of wireless and wired network, resulting in faster reaction to link layer loss. This scheme violates the end-to-end semantics of TCP. In I-TCP, it is possible the sender receives an acknowledgement of a data packet while the data packet has not reached the destination rather is buffered at the base station. However, many applications such as FTP and HTTP use application layer acknowledgements in addition to end-to-end TCP acknowledgements. Using I-TCP for these applications does not comprise end-to-end reliability.

Super TCP [10]. Because satellite channel is a FIFO channel, out-of-order routing and congestion on the satellite link are impossible. Super TCP uses one duplicate ACKs to trigger the retransmission at the base station and a fixed window size for the satellite TCP connection. It also proposes a new sender algorithm using the same idea as in TCP new Reno. It uses partial ACKs to calculate the burst loss gap and sends all the potential loss packets beginning from the partial acknowledgement number. It is possible that the sender could retransmit packets that have already been correctly received by the receiver.

SCPS-TP [19]. Space communication protocol standards-transport protocol (SCPS-TP) is a set of TCP extensions for space communications. This protocol adopts the Timestamps and window scaling options in RFC1323. It also uses TCP Vegas low-loss congestion control mechanism. SCPS-TP receiver doesn't acknowledge every data packet. ACKs 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. Because there is no regular acknowledgement-driven clock, it uses an open-loop rate control mechanism to meter out data smoothly. To transmit data continuously in the presence of link layer loss rather than congestion loss is especially important.

SCPS-TP uses selective negative acknowledgement (SNACK) to address this problem. SNACK is a negative acknowledgement and it can specify a large number of holes in a bit-efficient manner.

IV. Motivation and basic idea

The related work above tries to solve some of the TCP problems in the satellite data network. But a scheme that solves all the problems does not exist yet and the proposed solutions may not work well in some other networks. In addition, all TCP solutions in the literature are not independent of each other. A better solution can combine some of them and come up with a new protocol for specific satellite networks. The problems of Internet over satellite problems are far from being solved. In this paper, we propose a scheme, which takes into account the characteristics and requirement of aeronautical satellite networks. Our scheme shows significant improvement of link utilization.

In the aeronautical satellite network as in Figure 1, the client (passenger) on the aircraft accesses an Internet server over bent pipe GEO satellite. We focus on the transport layer protocol design and assume the point-to-point link between ground gateway and satellite gateway. The terrestrial link between the server and the ground gateway is actually a path through routers in the Internet with typical Internet delay and very low bit error rate. The aircraft link between the client and the aircraft gateway are wired/wireless LAN connection with very small delay and very low bit error rate. In order for this hybrid TCP/IP network to be commercially deployable, it must seamlessly interoperate with existing TCP/IP networks. The following two requirements must be satisfied. First, both the Internet servers and clients on aircraft must use standard TCP/IP protocol. Most of the passengers (especially business travelers) want to connect to the communication network with their own equipments (such as laptops, PDAs) because they are used to from their daily life. Also it is not possible to implement the new protocol at every server in the world. Second, we need to provide high utilization of the satellite channel. 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.

As stated in section II, 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 it is difficult for satellite TCP connections to actually operate with large windows.

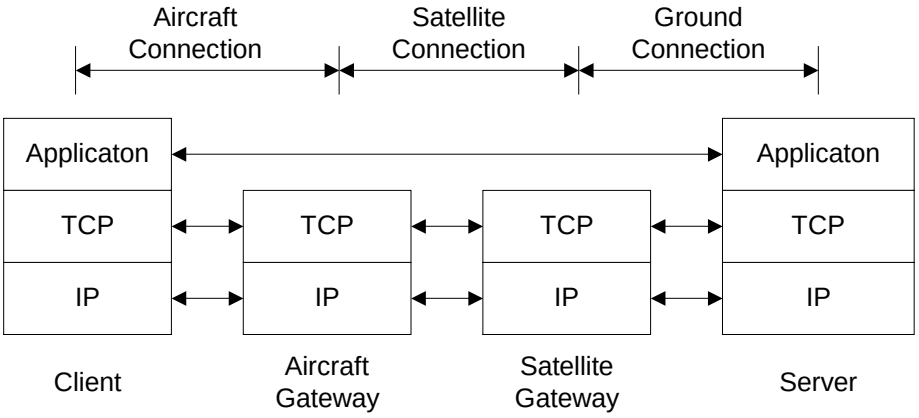

Figure 2: Protocol stack for a split connection configuration

Because the feedback information of the satellite 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 between the satellite gateways and aircraft gateway. Considering the interoperability issue, we propose a connection splitting based scheme. 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 configuration, 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, the second one is from the ground gateway to the aircraft gateway, and the third 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.

One advantage of the split connection approach is that it separates the losses on the satellite link from the losses on the Internet, allowing local recovery of lost packets. Another advantage of the split connection approach is that it allows tailoring the TCP implementation on each of the connections to best suit the characteristics of the underlying channel. The disadvantage is that the splitting scheme violates the end-to-end semantics of TCP, same as in I-TCP. However, application layer acknowledgements for most of the applications will not comprise end-to-end reliability.

V. TCP splitting Protocol

A. Connection Splitting

From above, the end-to-end TCP connection between the client on aircraft and the Internet server on ground is split into three connections: the ground connection between the Internet service and the ground gateway, the satellite connection between the ground gateway and the aircraft gateway, the aircraft connection between the client and the aircraft gateway. Both the ground connection and the aircraft connection have much large link bandwidth with very low bit error rate. The satellite link is the bottleneck. A satellite optimized transport protocol can be used for satellite TCP connection, while standard TCP protocol are used for other two connections. In this way, high utilization of the satellite link can be achieved, while there are no changes to the protocol stacks at the end users.

The advantage of splitting the TCP connection is that the satellite channel is isolated from the rest of the Internet. This channel has two unique properties, which differentiate it from the rest of the Internet. The first property is that packets sent on the satellite channel cannot be routed out of order. The second property is that congestion is not possible and therefore the only reason for packet losses is transmission errors. Both properties are attributable to the fact that there does not exist any router on the channel between the ground gateway and the aircraft gateway.

The above observations motivate us to 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 retransmit and to use a fixed window size for the satellite TCP connection. This implementation of AeroTCP will be discussed in details as follows.

B. Flow Control and Buffer Allocation

TCP uses flow control to ensure that the sender will never overflow the receiver's buffer. For every TCP connection (only one connection is shows in Figure 3), all packets waiting for transmitting or received are buffered at the send buffer or receive buffer, respectively. Consider the traffic from the Internet server to the client on the aircraft, all the TCP packets received from the server are forwarded to the TCP received buffer of the ground connection and they are moved from the receive buffer to the send buffer in sequence at the satellite gateway. Then the packets are sent from the send buffer to the aircraft gateway over the satellite link. At the aircraft gateway, the packets are moved from receive buffer of satellite connection to send buffer of aircraft connection in sequence. Finally, the packets are sent from the send buffer to client over aircraft connection. The return channel is only for ACKs.

Flow control is done between the satellite gateway and the aircraft gateway at the transport layer by using the receiver window, *rwnd*. For each satellite TCP connection, the aircraft gateway advertises a receiver window based on the available receive buffer space for that connection just as in TCP. Window scaling can be used here to advertise large windows. At the ground gateway, when packets are moved from ground connection receive buffer to satellite connection send buffer in sequence, a blocking write is performed so that the send buffer will not overflow. For ground TCP connection and aircraft TCP connection, standard TCP flow control is used so that the sender will not overflow the receiver's buffer. In this way, the traffic load at the satellite connection is back pressured to the receive buffer of the ground connection. When the traffic load on the satellite connection increases, the buffer begin to be filled up and a smaller receive window is going to be sent to the server. When the traffic load decreases, the

buffers begin to be emptied faster and larger advertised receiver windows are sent to the server so the server can speed up.

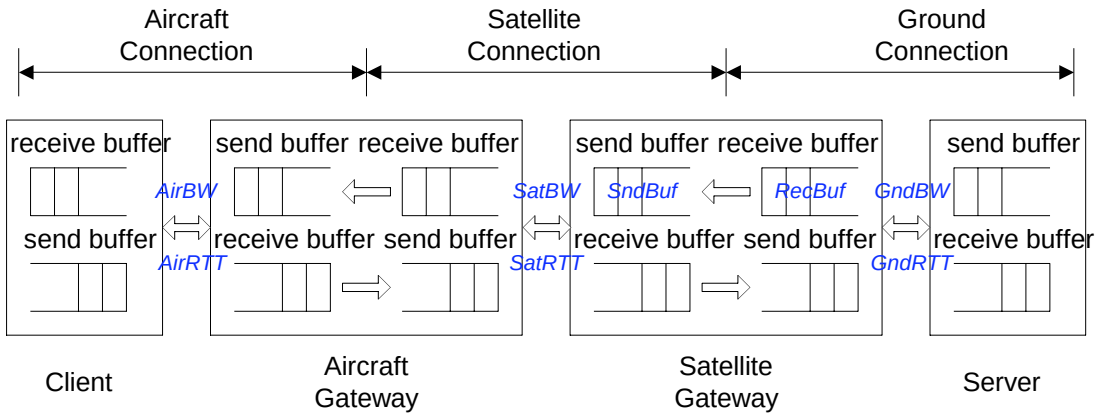

Figure 3: TCP flow control and buffer allocation

The buffer size assigned to each connection at the satellite gateway and aircraft 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. Based on the observation that the number of TCP connections is small at the client and the server compared to that at the gateways, we assume large buffer is available for each TCP connection at those nodes. Following we will discuss the buffer allocation at the ground gateway and the aircraft gateway.

Assume that the traffic is from Internet server to the client on aircraft. The bandwidth of the satellite link is much larger than those of the ground link and the aircraft link. Also assume there is only one connection in this system and the satellite link is error free. At the ground gateway, the send buffer of the satellite TCP connection is *SndBuf* and the receive buffer of the ground TCP connection is *RecBuf*. The effective satellite bandwidth, which is the raw satellite bandwidth deducted by the protocol headers, is *SatBW*. The round trip time for the satellite connection is *SatRTT* and for the ground connection is *GndRTT* (refer to Figure 3). When 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. The maximum achievable throughput of the end-to-end connection is $Throughput_{\max} = \min(SatBW, \frac{SndBuf}{SatRTT}, \frac{RecBuf}{GndRTT})$ [20]. 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 (BDP). However when the buffer size is greater than the bandwidth delay product, that is $SndBuf >= SatBW * SatRTT$, $RecBuf >= SatBW * GndRTT$, there are packets backlogged at the satellite gateway and these backlogged packets cannot contribute to the throughput and only increase the queuing delay. The same analysis applies to the aircraft gateway.

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. Based on the observation that although the average number of active connections is large, the variance is small. The bandwidth available to each connection does not vary dramatically. For simplicity, we assign each connection a static peak rate (PR), which is the maximum bandwidth it can achieve and is much smaller than the total satellite bandwidth. The buffer size is set to peak rate delay product (PRDP).

When the satellite link is error free, the buffer sizes allocated above are enough to achieve the target peak rate. However when the satellite link is not error free, changes need to be made at both ground gateway and aircraft gateway. When a packet is corrupted, the aircraft gateway has to buffer the out of order packets because the receiver of the satellite TCP connection only forwards in sequence packets. In order to keep the advertised receiver window open so that the sender of satellite connection can send new packets during the error recovery, the aircraft gateway needs a buffer size larger than the peak rate delay product (PRDP) to achieve the peak rate. The error control algorithm (will be discussed below) can recover multiple packet losses within one window in one RTT. If the error rate of the satellite link is low so that corrupted packets can be recovered in one RTT, receive buffer size about two times of the peak rate delay product should be provided. If the error rate is very high, retransmissions of the

corrupted packets can be lost again. In this case, receiver buffer size should be about three to four times of the peak rate delay product. The same buffer allocation scheme should be used for send buffer of satellite TCP connection. However, for the send buffer at the aircraft gateway, only one peak rate delay product is enough since the aircraft TCP connection has very short delay and very low bit error rate. For the same reason, the receive buffer at the ground gateway only need one peak rate delay product buffer size to achieve the target transfer rate. However, we set this buffer to twice of peak rate delay product. This is because 1), the ground link has some round trip delay and low bit error rate. Although they use standard TCP protocol, large buffer can ensure the data flow when there are packet losses. 2), we want the server to send little more data packets to the satellite gateway so that there are some packets backlogged at this buffer. Whenever the satellite TCP connection recovered from packet losses, it will not get starve for new packets to send.

C. Congestion Control and Bandwidth Allocation

The above flow control and buffer allocation scheme can ensure that each TCP connection will be able to achieve its target peak rate. However, without additional congestion control algorithm, it cannot guarantee network stability and fairness among TCP connections. There are two reasons: 1) It is difficult to allocate the buffer size for each TCP connections based on link conditions and number of connections because the link bit error rate is difficult to measure or estimate accurately. Thus the safest way is to allocate large buffer size for each connection. If there is no error or very low error rate in satellite channel, this buffer allocation will *temporarily* cause each TCP connection at ground gateway to send packets 2~4 times of peak rate to lower layers and overflow the IP buffer or the physical layer queue. 2) The peak rate is set based on the measurements of the traffic characteristics and the target satellite link utilization. The buffer size corresponding to that peak rate is set when the connection is initialized. They are difficult to change for active connection when the number of connections increases or decreases. Thus the buffer allocation could cause problems such as underutilization of the satellite link, unfairness and un-scalability.

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. In our aeronautical networks, 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 addition, congestion is not possible for the satellite connection because of the splitting setup. In our scheme, we cancel all the congestion control algorithms in TCP. However congestion window is still used to guide the transmission to ensure the network stability and fairness.

If there is only one connection, this TCP can send packets to IP layer at the rate corresponding to the satellite bandwidth. The congestion window can be fixed as satellite bandwidth delay product. As long as there are packets in the send buffer of the ground gateway and the receiver's window allows, the satellite link can be fully utilized. However, when there are multiple connections, without a fair queuing scheduler, the congestion window can be allocated by using the static bandwidth allocation and adaptive bandwidth allocation scheme.

The simplest scheme of buffer allocation is static bandwidth allocation. Each TCP connection is assigned the same congestion window. If there are N connections in the system with satellite bandwidth of SatBW, each connection is allocated SatBW/N to ensure the fair sharing. The congestion window is set as SatBW/N*SatRTT. The conservative method here is to use the maximum number of connections in that equation. However, when the number of connection is much smaller than the maximum number, the satellite link is underutilized.

The congestion window, thus the bandwidth allocation for each connection, can be set adaptively. To achieve fairness, each connection will need SatBW/N*SatRTT for its congestion window. This window size will increase when the number of connection N decrease. When some connections are closed, the total number of connection is decreased. Other TCP connection will increase its congestion window and speed up its transmission rate. However, this scheme requires that the number of connections is small and changes very slow such that other connections could catch up with the change in time. This is only suitable for video or ftp applications with long connection duration. Another adaptive bandwidth allocation is set the congestion window based on the measurements of the traffic characteristics and the target satellite link utilization. The target bandwidth for each connection may be changed from time to time, but it could be measured and stored in some predefined table. Whenever a new connection is initialized, it could set its congestion window by looking up the table. In this way, we can guarantee fair bandwidth sharing without a fair queuing scheduler.

Both those bandwidth allocation scheme have some drawbacks and limitations. The more advanced bandwidth allocation and flow control schemes will be discussed in future works.

D. Super Fast Error Control

The satellite link in Ka band is often characterized by high bit error rate and burst losses since the weather conditions greatly affect link availability. The burst losses normally cause TCP retransmission timeouts and significant throughput degradation. In this section, we first discuss the behavior of TCP Reno and TCP New-Reno in the presence of burst losses and then present our super fast error control algorithm for AeroTCP. This error control algorithm can recover a loss of multiple packets in a burst in one RTT while Reno and New-Reno can only retransmit at most one dropped packet per RTT.

TCP depends on duplicate acknowledgements and timer for error control. As we discussed in section II, the TCP Reno does not perform well under burst losses. The fast retransmit algorithm is triggered after three duplicate acknowledgements are received. In order for the fast retransmit algorithm to recover the loss, the congestion window size has to be greater than four for single packet loss and has to be greater than ten for two consecutive losses in one window. While for three or more consecutive losses in one window, the TCP sender has to wait for timeout to recover the loss.

A modified version of TCP, called TCP New-Reno, aims at avoiding timeouts when multiple packets are lost in the same window by changing the behavior of the TCP sender during fast retransmit as follows. The protocol defines a fast retransmit phase as the time between the receipt of 3 dup ACKs and the time when an ACK arrives for all the packets that were outstanding when the phase started. This is in contrast to the regular Reno implementation, where the fast retransmit mode lasts until the ACK for the retransmitted packet arrives. When multiple packets are lost in the same window, the retransmission of the first lost packet triggers a partial ACK, an ACK that acknowledges some but not all the packets what were outstanding at the start of fast retransmit. A partial ACK is treated as a signal that the packet whose sequence number is indicated has been lost and should be retransmitted. In this pattern, the TCP New-Reno can recover one lost packet during each RTT.

In AeroTCP, we explore the specific characteristics of the satellite network. Firstly, The congestion is impossible for the satellite connections and any loss much be caused by the link layer corruption. 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 propose a new error control algorithm based on SACK TCP. The basic idea is to use one dup ACK for fast retransmit, use fixed window for satellite connection, and recover packet losses based on SACK information.

In SACK TCP, the SACK option field contains a number of SACK blocks, where each SACK block reports a non-contiguous set of data that has been received and queued. The first block in the SACK option is required to report the data receiver's most recently received segment, and the additional SACK blocks repeat the most recently reported SACK blocks. In our research each SACK option is assumed to have room for three SACK blocks. The congestion control algorithms implemented in the SACK TCP are a conservative extension of Reno's congestion control, in that they use the same algorithms for increasing and decreasing the congestion window. However, in AeroTCP, we change the congestion control algorithm as discussed in previous section. The AeroTCP implementation preserves the properties of TCP SACK of recovery multiple packet losses from one window of data and uses retransmit timeouts as the recovery method of last resort.

In AeroTCP, the sender enters super fast recovery when it receives one duplicate acknowledgement since out-of-order delivery in satellite channel is impossible. The sender retransmits a packet but will not cuts the congestion window in half. During fast recovery, the sender maintains a variable called *pipe* that represents the estimated number of packets outstanding in the path. The sender only sends new or retransmitted data when the estimated number of packets in the path is less than the congestion window. The variable *pipe* is incremented by one when the sender either sends a new packet or retransmits an old packet. It is decremented by one when the sender receives a dup ACK packet with a SACK option reporting that new data has been received at the receiver.

Use of the *pipe* variable decouples the decision of when to send a packet from the decision of which packet to send. The sender maintains a data structure, the *scoreboard*, which remembers acknowledgements from previous

SACK options. When the sender is allowed to send a packet, it retransmits the next packet from the list of packets inferred to be mission at the receiver. If there are no such packets and the receiver's advertised window is sufficiently large, the sender sends a new packet.

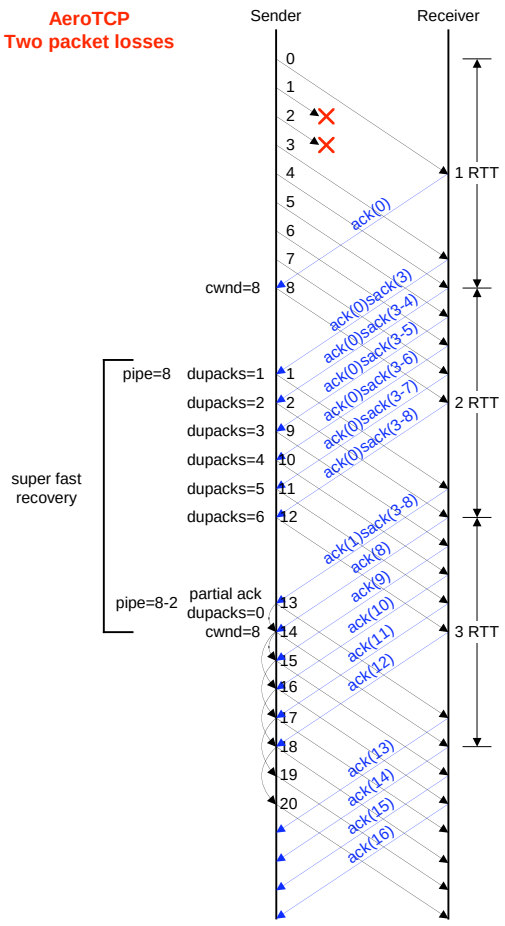

Figure 4: AeroTCP with 2 packet losses

The sender exits super fast recovery when a recovery acknowledgement is received acknowledging all data that was outstanding when fast recovery was entered. When a retransmitted packet is itself dropped, the AeroTCP implementation detects the drop with a retransmit timeout. However, the timer has a finer granularity. After timeout, two copies of the dropped packet are sent to increase redundancy.

The AeroTCP sender has special handling for partial ACKs. For partial ACKs, the sender decrements *pipe* by two packets rather than one. When super fast recovery is initiated, *pipe* is effectively decremented by one for the packet what was assumed to have been dropped, and then incremented by one for the packet what was retransmitted. However, for partial ACKs, *pipe* was incremented when the retransmitted packet entered the pipe, but was never decremented for the packet assumed to have been dropped. Thus when a partial ACK arrives, it does in fact represent two packets that have left the pipe: the original packet (assumed to have been dropped), and the retransmitted packet.

Figure 4 shows the sequence of events for AeroTCP with two packet losses. At the beginning, packets 1-8 are sent while packets 1 and 2 are lost. After receiving the first ACK for packet 0, the sender goes into the super fast recovery, retransmits the lost packet 1, and initializes the *pipe* as *cwnd*. The second dup ACK causes the value of *pipe* to become 7 and contains SACK information, indicating a hole at packets 2. Packet 2 is retransmitted. Next 4 dup ACKs allow 4 new packets to be sent. From *scoreboard*, no holes remain to be filled and the sender may send new packets 9-12. The next ACK arrives corresponding to the receipt of retransmitted packet 1. It is a partial ACK, causing *pipe* to be decremented by two and allowing the sender to send packets 13-14 (packet 14 is scheduled to be sent in next time slot). The next ACK received corresponds to the receipt of retransmitted packet 2 and brings the sender out of super fast recovery with a congestion window of 8. After packet 20, the TCP returns back to its normal sliding window transmission pattern.

VI. Simulation Results

To test our splitting scheme, we setup a simulation scenario as shown in figure 1. A client downloads files using FTP from a server via a GEO satellite. The aircraft and the satellite will track each other during a domestic flight (Assume the GPS information is available at those nodes). The round trip delay between the ground gateway and aircraft gateway is about 480ms while the round trip delay for the ground connection is about 80ms. The satellite link bandwidth is about 400Kbps for one connection. The bit error rate of satellite channel is computed based on the signal-to-noise ratio at the receiver. We add a two state Gilbert-Elliot model before the receiver to simulate the burst losses. The ground link and aircraft link have a bandwidth of 100Mbps without link error. The transport protocol for the ground connection and aircraft connection are normal TCP SACK, while we use AeroTCP for the satellite connection. The statistics we collect is link utilization and application response time. We compare three scenarios in our simulation: our AeroTCP, TCP splitting (with standard TCP SACK for satellite link), and End-to-End TCP SACK.

Figure 5 shows the satellite link utilization for three schemes. There are 10 TCP connections with total bandwidth of 4Mbps. Each TCP connection downloads a file of 1.6M Bytes. The figure shows the aggregate utilization for all connections. 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, while end-to-end TCP has little difficult to increase its window fast enough to fully utilize the satellite bandwidth. 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. For End-to-End TCP solution, the performance gets even worse when the bit error rate increases. 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.

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}. We get similar results or more obvious results when we use large FTP files. It also shows the improvement for other applications like HTTP, Email, video and audio data.

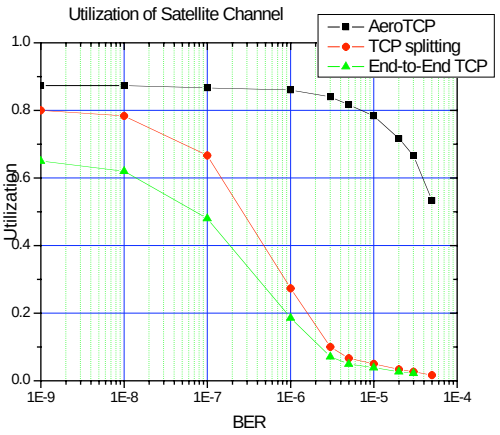

Figure 5 Utilization for different bit error rates

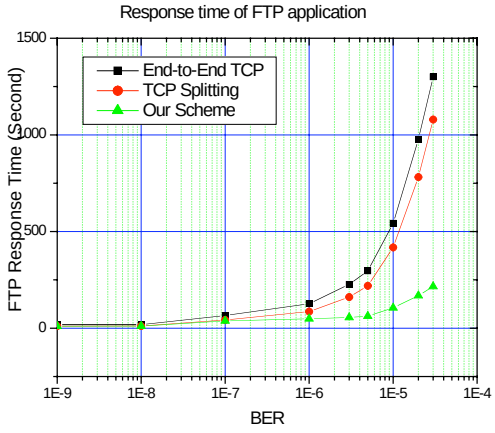

Figure 6 Response time for FTP application

VII. Conclusion and future works

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. 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.

A common way to characterize the performance of an access network is in terms of the throughput observed by the applications running over the TCP/IP protocols. The throughput achieved depends on three facts: the bandwidth allocation scheme for different applications and users, the packet loss rate, and the specific TCP/IP implementation. In this paper, we focus on the transport protocol and present a new solution for aeronautical network. However, in our study, a distributed bandwidth allocation scheme is used as in section 5. We also assume that there are only FTP

or HTTP traffic in our system with same priority. In aeronautical networks, there are more other applications and services with different quality of service (QoS) requirements. The passengers' communication will compete the bandwidth with high priority applications for airspace system operations. The advance bandwidth allocation and flow control schemes should be used here to guarantee the high utilization of satellite channel, the QoS for different traffic sources, and the fairness between different users. We propose to implement a priority queue in IP layer for each types of traffic source. A centralized scheduler will be used to allocate the bandwidth for different sources. A rate based flow control scheme will be used to guide TCP connections to send data to IP queue. This flow control scheme can remove the drawback of the congestion control and bandwidth allocation scheme in this paper and improvement the throughput. This will be the direction of future works.

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 Boeing[SM]-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]C. Partridge and T. Shepard, "TCP performance over satellite links", IEEE Network, 11:44-49, September 1997.

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

[10]I. Minei and R. Cohen, "High-speed Internet access through unidirectional geostationary satellite channels", IEEE J. Select. Areas Comm., 17, February 1999

[11]S. Floyd and T. Henderson, and A. Gurtov, "The New Reno Modification to TCP's Fast Recovery Algorithm", Internet RFC 3782, April 2004.

[12]J. Mahdavi, S. Floyd, M. Mathis, and A. Romanow, "TCP Selective Acknowledgements Options", RFC 2018, April 1996.

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

[14]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

[15]R. Braden, "T/TCP-TCP extensions for transactions, functional specification", RFC 1644, 1994.

[16]M. Mathis and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control", in Proc. ACMSIGCOMM, pp. 281-291, Aug. 1996.

[17]S. Keshav and S. Morgan, "SMART retransmission: performance with overload and random losses", in Proc. IEEE Infocom, pp. 1131-1138, Apr. 1997.

[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]R. C. Drust, G. Miller, and E. J. Travis, "TCP extensions for space communications", Proc. ACM Mobicom, '96, Nov. 1996.

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