TCP over GEO satellite hybrid networks

Xiaoming Zhou
John S. Baras

Institute for Systems Research and
Center for Satellite and Hybrid Communication Networks
University of Maryland College Park, MD 20742

ABSTRACT

Broadcast satellite networks are going to play an important role in the global information infrastructure. Satellites can provide direct-to-user Internet services (i.e. DirecPC from Hughes Network System and several other systems) and they can also serve as traffic trunks in the middle of the network. About 98 percent of the Internet traffic is TCP traffic. TCP works well in the terrestrial fiber networks. However TCP does not work well in satellite and in hybrid (satellite-terrestrial) networks. In this paper we analyze the problems that cause this dramatically degraded performance and review the solutions proposed to date in the literature. Based on the observation that it is difficult for an end-to-end solution to solve these problems in this kind of hybrid network, we propose a connection splitting based solution. While a lot of research has been done on improving TCP throughput for a single connection, we consider the case that multiple connections with different round trip time compete with each other. The crucial part of our scheme is the flow control algorithm at the hybrid gateways (i.e. the interfaces between the satellite network and the terrestrial network). Because the end-to-end connection is split into two at the hybrid gateways, the data packets are buffered at the TCP layer rather than at the IP layer as in a normal router. Furthermore the TCP layer not only buffers data waiting for transmission, but also buffers data transmitted but not yet acknowledged. The flow control algorithm should avoid stall caused by buffer exhaustion, guarantee fairness among all competing connections and maintain high utilization of the satellite link. We demonstrate via performance evaluation that the scheme we propose meets these requirements.

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 network exploits three observations [1]: 1) some rural area may not be reached by fiber network or it may be too expensive to do so 2) satellite hybrid network can provide higher 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.

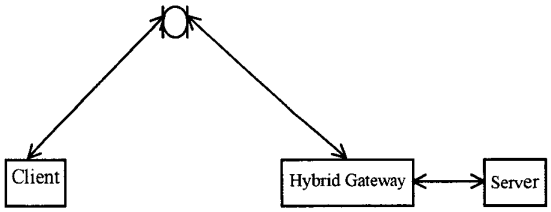

Figure 1: Direct to User satellite hybrid network

Satellites can also be in the middle of the data networks to provide communication between two gateways located geographically far away from each other [2] (Figure 2).

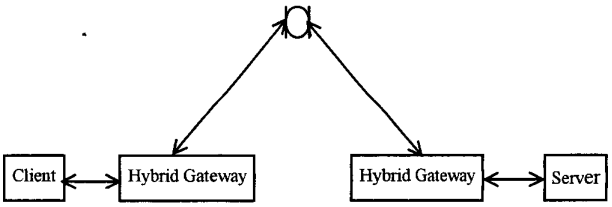

Figure 2: satellite in the middle network

GEO satellite is about 36,000km above the earth. The propagation delay between the ground terminals and the satellites is about 125ms. Therefore a typical round trip time (RTT) for two-way system is about 580ms including about 80ms RTT for the terrestrial networks. The time TCP spends in slow start equals $RTT*\log_2 SSTHRESH$[1] when every segment is acknowledged and equals $RTT*\log_{1.5} SSTHRESH$ [3] when every other segment is acknowledged. For a connection with large RTT, it spends a long time in slow start before reaching the SSTHRESH. 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 [4] (i.e. 4 MSS) rather than one for slow start. So files less than 4K bytes can finish its transfer in one RTT rather than 2 or 3. Another proposal [5] is to cancel the delay acknowledgement mechanism in the slow start so every

[1] SSTHRESH – slow start threshold

0-7803-7625-0/02/$17.00 ©2002 IEEE.

packet get acknowledged and the sender can increase its congestion window (CWND) more quickly. For bulk transfer, TCP throughput is inverse proportional to RTT [6]. So TCP connection with larger RTT does not get its fair share of the bandwidth when it competes with the connections with smaller RTT. Using simulations, Henderson claims the 'Constant-rate' additive increase policy can correct the bias against connection with long RTT [7]. However it is difficult to implement this policy in a heterogeneous network.

The bandwidth delay product in this system is very large. In order to keep the pipe full, the window should be at least the bandwidth delay product [5]. However the receiver advertised window that is 16 bits in the TCP header cannot be more than 64k, which limits the two-way system throughput to 64k/580ms=903Kbps. Window scaling [8] is used to solve this problem. However when the window is large, it is quite possible for multiple losses in one window, which leads to poor performance. For the same reason, the sender buffer can also limit the TCP connection throughput if it is less than the bandwidth delay product, which is usually the case.

Ka band satellite channel is noisier than fiber channel. Bit error rates of the order of 10^{-6} are often observed [9]. Because TCP Reno treats all losses as congestion in the network, this kind of link layer corruption can cause TCP to drop its window to a small size and leads to poor performance. Forward error correction (FEC) coding is usually used in satellite communication to reduce the bit error rate (BER). FEC can reduce the BER to 10^{-10} for about 99.5% of the time for the terminals and 99.75% of the time for the gateways. 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. TCP SACK [10] is proposed to convey non-contiguous segments received by the receiver in the ACKs so that the sender can recover error much faster than TCP Reno, which well known can recover only one loss per RTT.

There are several solutions proposed in the literature to solve these problems. TCP peach [11] 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 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. 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 priority mechanism, which makes it difficult to deploy.

Space communication protocol standards-transport protocol (SCPS-TP) [12] is a set of TCP extensions for space communications. This protocol adopts the Timestamps and window scaling options in RFC1323 [8]. 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. Because there is no regular acknowledgement-driven clock, 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.

Because satellite channel is a FIFO channel, there is no out-of-order routing. And congestion on the satellite link is impossible if the packets are sent at the rate of the satellite bandwidth. Reference [13], a connection splitting based solution, proposes to use one duplicate ACK to trigger the retransmission at the hybrid gateway (HGW) and to use a fixed window size for the satellite TCP connection. The paper proposes a new sender algorithm using the same idea as in TCP new Reno [14,15]. It uses partial ACKs to calculate the bursty loss gap and sends all the potential loss 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 was shown that this algorithm performs better than TCP SACK in recovering bursty errors.

Flow control scheme at hybrid gateways

All these above proposals are not independent of each other. A better solution can combine some of them and come up with a new protocol for satellite network. All in all, the new protocol should decouple the congestion control from error control and fill the long-fat pipe with enough data packets to improve throughput. For an end-to-end scheme such as window scaling, the large window of packets need to be kept in the sender buffer, which means that the server need to allocate larger buffer for satellite connections. It is difficult to upgrade all the Internet servers to be satellite friendly in a short time period. Satellite connections cannot get their fair share of bandwidth when they compete with connections of smaller RTT. It is also difficult for end-to-end solutions to solve the fairness problem. An alternative to end-to-end scheme is to keep the large window of packets in the network such

as the hybrid gateway between the satellite and terrestrial networks. Considering the interoperability issue, we propose a proxy-based scheme and design a flow control algorithm for the proxy to couple multiple terrestrial connections and satellite connections together while guarantees fairness among connections and maintains high utilization of the satellite link. We assume the satellite link is the bottleneck of the system and the terrestrial network has enough bandwidth. We also assume a strong FEC is used for the satellite link so the error is handled at the link layer. We use the direct-to-user topology for our simulations and focus on the flow control scheme at the hybrid gateway. It is straightforward to extend the scheme to the traffic trunk topology.

An end-to-end TCP connection is split into two connections at the hybrid gateway. One connection is from the TCP server to the hybrid gateway and another one is from the hybrid gateway to the TCP receiver. 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. Hybrid gateway sends premature acknowledgements to the Internet servers and takes responsibility to relay all the acknowledged packets to the TCP receivers reliably. In our analysis and simulations, we consider FTP traffic only.

A. Queuing model at the hybrid gateway

Because TCP connection splitting is used at the hybrid gateway, the data is buffered at the TCP layer rather than the IP layer. For a normal router, only those packets waiting for transmitting are buffered at the IP layer. However, the hybrid gateway has to buffer the packets waiting for transmission as well as packets that have been transmitted but not acknowledged. A normal router keeps all the packets in a FIFO queue while the hybrid gateway has a queue for each TCP connection.

All the TCP packets received from the servers are forwarded to the TCP layer receive buffer of the SERVER-HGW connection and data moved from the receive buffer to the send buffer of the HGW-CLIENT connection. Then the packets are sent from the send buffer to the IP layer. From Figure 3, we can see that the IP input queue should be empty if we assume the processing rate of the hybrid 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 pointer. The queuing model at the hybrid gateway can be simplified as in Figure 3, in which the receive buffer and the send buffer are represented by one buffer.

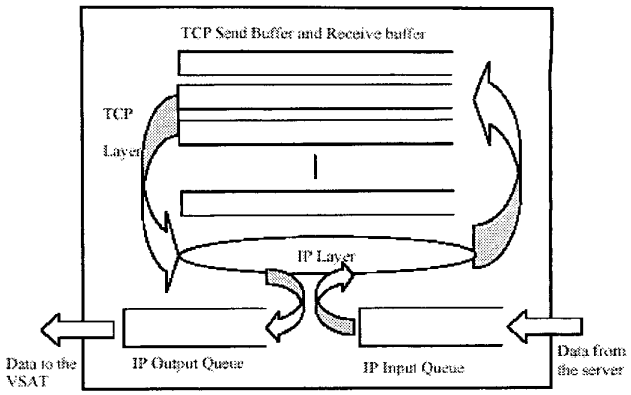

Figure 3 Simplified queuing model at hybrid gateway

B. Single connection case

The buffer size assigned to each connection at the hybrid gateway has a direct impact on the end-to-end TCP throughput. First we assume there is only one connection in this system, we consider multiple connections case in the next section. The buffer size assigned to the connection is Buff and the effective satellite bandwidth[2] is SatBW, which is known and is a constant. 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 HGW should be equal to the output rate of the queue, i.e. RecvWin/TerrRTT = SatWin/SatRTT (1).

Case1: Buff < SatBW*(SatRTT+TerrRTT)

Buff = RecvWin + SatWin (2)

From (1) and (2), we can get SatWin = Buff*SatRTT/(TerrRTT + SatRTT) and the throughput of the satellite TCP connection is SatWin/SatRTT = Buff/(TerrRTT+SatRTT) which is smaller than SatBW. This means that the buffer size becomes the bottleneck of the system and the SatBW cannot be fully utilized if Buff is smaller than SatBW*(SatRTT+TerrRTT).

Case 2: Buff = SatBW*(SatRTT+TerrRTT) (3)

From (1) (2) and (3), we can get SatWin = SatBW*SatRTT and the throughput of the TCP connection is SatWin/SatRTT i.e. SatBW. This means that the TCP connection can achieve the satellite effective bandwidth.

[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 on the satellite link.

This corresponds to the state that all the packets are flowing in the links and no backlog packets are in the HGW queue.

Case 3: Buff > SatBW*(SatRTT+TerrRTT) (4)
SatWin/SatRTT cannot be larger than SatBW, i.e. SatWin <= SatBW*SatRTT. For the same reason, TerrWin <= SatBw*TerrRTT. Therefore the throughput of the TCP connection is SatBW and there are Buff – SatBW*(SatRTT+TerrRTT) backlog packets buffered at the hybrid gateway. These backlog packets cannot contribute to the throughput and they only increase the queuing delay.

From the above analysis, the throughput of the connection is MIN(SatBW, Buff/(SatRTT+TerrRTT)) and the backlog packets are MAX(0, Buff - SatBW*(SatRTT+TerrRTT)).

C. Multiple connections case

For the multiple connections, the round trip time of the HGW-CLIENT connections is almost the same, however the round trip time of the SERVER-HGW connections could have a lot of difference because the servers are connected with hybrid gateway through wide area networks.

1. Static buffer allocation

The simplest scheme of buffer allocation is to allocate all the connections the same buffer. Suppose there are 2N connections in this system, i.e. N SERVER-HGW connections and N HGW-CLIENT connections.

Notations:
FairShare -- the fair share of the satellite bandwidth, which is SatBW/N
Buff -- buffer allocated to a single connection
MaxRTT -- MAX ($SatRTT_i + TerrRTT_i$), $1 \le i \le N$
MinRTT -- MIN ($SatRTT_i + TerrRTT_i$), $1 \le i \le N$

Case 1: Buff < FairShare*MinRTT
This buffer allocation corresponds to case 1 in last section. No connections can achieve its fair share. Therefore the utilization of the satellite link is low. We call this system buffer bottlenecked rather than bandwidth bottlenecked.

Case 2: FairShare*MinRTT ≤ Buff ≤ FairShare*MaxRTT
This allocation ends up with some connections cannot achieve their fair share because they are buffer bottlenecked while others can achieve more than its fair share. This scheme leads to unfairness among competing connections. The connections with smaller SERVER-HGW RTT can grab more bandwidth.

Case 3: Buff > FairShare*MaxRTT
All connections can achieve its fair share and all of them have some backlog packets at the hybrid gateway. Note that the FairShare is not fixed and is inverse proportional to N. The trivial scheme may be to set Buff to (SatBw/MIN(N))* MaxRTT, where MIN(N) is the minimum number of connections the system will encounter and the MaxRTT is the maximum end-to-end RTT of all possible connections. This scheme is too conservative and causes a lot of backlog packets buffered at the HGW. Furthermore the buffer needed at the HGW increases linearly with the number of connections. This makes this scheme not scalable when the number of active connections is much larger than MIN(N). If the total memory available at the HGW has a limit, which is usually the case, it is possible that newly arrived connection is rejected or queued because there is not enough buffer at the HGW.

2. Adaptive buffer allocation

The static buffer allocation could cause problems such as underutilization of the satellite link, unfairness and un-scalability. Observe that the buffer needed for a connection i to achieve its fair share is (SatBW/N)*($SatRTT_i$ + $TerrRTT_i$) which decreases when N increases. This motivates us to allocate buffer to the connections dynamically based on the number of connections in the system and the round trip time of the connections [16]. When the HGW receive a SYN packet, it increases N. However it does not decrease the buffer size of active connections immediately, rather it decreases the buffer size only when packets are acknowledged by the clients. If the buffer size is decreased by the size of all the packets acknowledged, this could stall the server. In order not to cause oscillation, our scheme decreases the buffer size by only half of the acknowledged data. This can reduce the input rate of the SERVER-HGW connections more gradually. When the number of connections in the system decreases, the fair share bandwidth SatBW/N increases. Therefore the buffer allocated to active connections should increase. When the HGW receive a FIN packet, it decreases the global variable N and increases buffer allocations to the active connections immediately. To bring a connection to reach its equilibrium point more quickly, HGW sends an acknowledgment to the server as soon as it receives an acknowledgement from the client. These acknowledgements serve as receiver window update to the servers and they can notify the servers about the buffer availability more promptly than the data driven acknowledgements of the SERVER-HGW connections.

Considering the effective satellite bandwidth and the number of active connections are known, there is no need to use slow start to probe the bandwidth on the HGW-CLIENT side. We cancelled all the congestion control algorithms of the TCP connections on the HGW-CLIENT side and connection i can start at a window (SatBW/N)* $SatRTT_i$. We will show in the next section that our scheme can guarantee fair bandwidth sharing without a fair queuing scheduler.

An easy extension to this is to provide weighted proportional fairness to different connections. Connection i with weight w_i can start at window SatBW* $(w_i/\sum w_i)$*$SatRTT_i$ and the buffer allocated to it should be SatBW* $(w_i/\sum w_i)$*($SatRTT_i$ + $TerrRTT_i$). This provides an easy way to provide different services to different users, e.g. users who pay more can get more bandwidth.

In stable state the above schemes can guarantee high utilization, fairness and scalability. However in the transient state, it could lead to low link utilization. When the number of connections in the system decreases, there is a period for the left connections to speed up. During this period, the satellite link is under utilized. In our scheme, the buffer allocated to connection i is (SatBW/N)*($SatRTT_i$ + $TerrRTT_i$) plus some BACKLOG packets. These packets are kept at the HGW to maintain high utilization. On the other hand, when the number of connections increases, the newly started connections need some time to ramp up to their fair share. The old connection i now has a smaller window (SatBW/N)*($SatRTT_i$+ $TerrRTT_i$) because of the larger N. In order to keep the satellite link busy, additional packets besides its window can be sent if the number of packets in the IP output queue is less than some threshold. After the newly started connections reach their fair share, the old connections begin to reduce their buffer size.

D. Simulation results

1. Single connection case

The topology is based on Figure 1 and only one connection is set up. The SatRTT is 500ms and the TerrRTT is 80ms. The maximum segment size of TCP is 512 bytes and the file size transfer from the server to the client is 4M bytes. The raw satellite bandwidth is 400kbps and the terrestrial bandwidth is 800 kbps. The buffer size set by our scheme is about 28K bytes. From Figure 4, we can see the throughput of the satellite link is about 400kbps and from Figure 5, we can see the packets are sent to the satellite link as soon as they arrive at the HGW because there is no backlog packets at the HGW. When the buffer size is increased from 28K to 34K, the throughput is the same and there are about 6K bytes backlogged in the HGW (Figure 5), which only increase the queuing delay. On the other hand, if we decrease the buffer size from 28K to 24K, the throughout is less than 350kbps and the system becomes buffer bottlenecked.

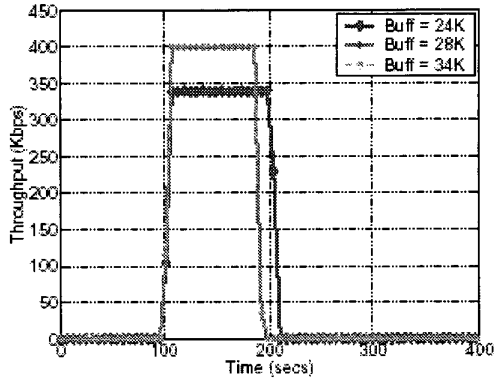

Figure 4 Satellite link throughput for different buffer sizes

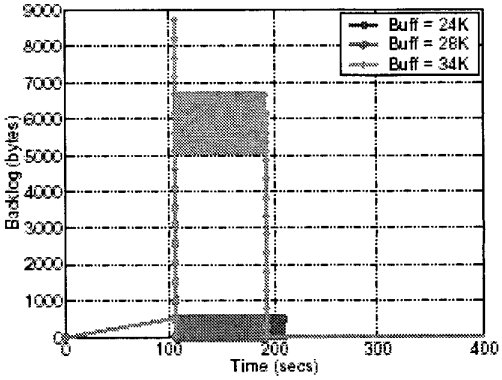

Figure 5 Backlog for different buffer sizes

2. Multiple connections case

For multiple connections, we set the satellite bandwidth to 6M bps. From Figure 6 we can see when there are ten connections in the system, each of them gets its fair share i.e. 600kbps. When the number of connections increases from 10 to 15, the throughput decreases to 400kbps. At about 200 sec, ten connections finish their transfer. The left five connections can ramp up to 1.2Mbps. In our simulations, we assume the CLIENT terminals have enough buffers, so the HGW-CLIENT connections are not receiver window limited. The BACKLOG is set to two packets. Figure 6 shows that adaptive buffer allocation can guarantee fairness and maintain high satellite link utilization.

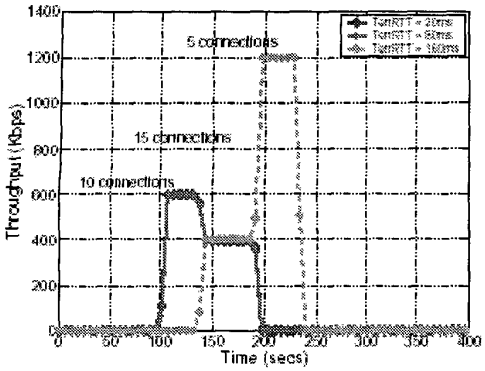

Figure 6 Throughput with adaptive buffer allocation

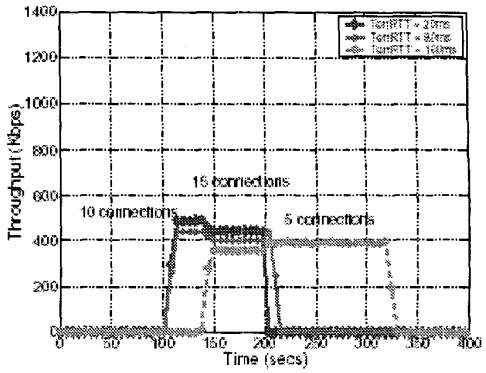

Figure 7 Throughput with static buffer allocation

The maximum buffer usage in the adaptive allocation scheme is about 456K bytes when there are 15 connections in the system. For comparison, each connection in the static scheme is allocated buffer size of 31K bytes so that the total buffer usage is the same when the system has 15 connections. From Figure 7, we can see that when there are ten or five connections in the system, it is buffer bottlenecked and the utilization of the satellite link is low. While when there are 15 connections in the system, the utilization ramps up, however it leads to unfairness. Connections with smaller RTT achieve higher throughout at the sacrifice of connections with larger RTT.

Conclusions and future work

Because it is difficult for an end-to-end solution to solve the problems in the satellite hybrid network, we purposed a proxy-based solution. A flow control algorithm at hybrid gateway adaptively allocates buffer to each connection based on the number of connections in the system and the round trip time. Through simulations, we show our scheme can maintain high utilization of the satellite link and almost perfect fairness among TCP connections. Furthermore, our scheme is more scalable than static buffer allocation scheme.

Because connection splitting needs to access the TCP header, it will not work if IPSEC is used. One possible way out is layered IPSEC technique. TCP header in packet is encrypted with one key, and the rest of the packet is encrypted with a different key. The hybrid gateway only has the key to decrypt the header.

Usually TCP traffic and multicasting UDP traffic share the satellite bandwidth, future work will consider the bandwidth sharing between TCP and UDP traffic. Therefore the bandwidth allocated to TCP traffic is not fixed rather dynamically changed with the traffic arrival pattern of the UDP traffic.

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. V. G. Bharadwaj "Improving TCP Performance over High-Bandwidth Geostationary Satellite Links" *Master Thesis CSHCN M.S. 99-7* http://www.isr.umd.edu/CSHCN
3. Partridge and T. Shepard, "TCP performance over satellite links," *IEEE Network*, vol. 11, pp. 44–49, Sept. 1997.
4. M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window," *Internet RFC 2414*, September 1998
5. M. Allman *et al.*, "Ongoing TCP research related to satellites," *RFC2760*, Feb. 2000.
6. 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
7. T. Henderson, E. Sahouria, S. McCanne, and R. Katz, "On improving the fairness of TCP congestion avoidance," in *Proc. IEEE GLOBECOM'98 Conf.*, 1998
8. V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," *Internet RFC 1323*, 1992
9. 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.
10. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," *Internet RFC 2018*, 1996.
11. 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
12. R. C. Durst, G. Miller and E. J. Travis, "TCP extensions for space communications," *Proc. ACM Mobicom*, '96, Nov 1996
13. I. Minei and R. Cohen "High-speed internet access through unidirectional geostationary satellite channels", *IEEE J. Select. Areas Commun.*, Vol. 17 Feb 1999
14. S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *Internet RFC 2582 (Experimental)*, April 1999.
15. J. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," in *Proc. ACM SIGCOMM*, Aug. 1996, pp. 270–280.
16. J. Mahdavi, J. Semke and M. Mathis, "Automatic TCP buffer tuning", *Computer Communication Review*, 28(4), October 1998.