Performance Improvement of TCP-based Applications in a Multi-access Satellite System

Vivek Mhatre and Catherine Rosenberg
School of Electrical and Computer Eng.
Purdue University
West Lafayette, IN-47906
Email: {mhatre, cath}@ecn.purdue.edu

***Abstract*— This paper describes a TCP splitting solution for multi-access bent-pipe GEO satellite networks. Such networks have a star topology with a large number of user terminals connected to a hub over a satellite link with broadcast downlink and multi-access uplink. In such systems, besides improving the end-to-end performance of TCP, efficient usage of the multi-access uplink is a key design objective. Most of the work done in the area of TCP over satellite so far has focused on TCP splitting in point-to-point satellite links. In this paper we first discuss the resource allocation issues involved in TCP splitting over multi-access satellite links. We then propose a TCP splitting scheme that uses Satellite Transport Protocol [13] and is well integrated with the MAC protocol. We propose an efficient and fair scheduling algorithm at the hub for uplink bandwidth allocation which works together with Bandwidth on Demand (BoD), wherein the terminals request for time slots to send data over the uplink. We mainly concentrate on improving the performance of HTTP like applications which are characterized by short bursty requests from clients to servers. We have developed the *ns* simulator to have several new modules that can be used to simulate a whole new range of satellite-terrestrial network topologies, transport level proxies, and TCP and STP flow control behavior.**

I. Introduction

DIGITAL satellites are extensively used for broadcasting entertainment services. Their role in delivering broadband access to the Internet is becoming increasingly important due to their high downlink capacity and large coverage. A first generation of satellite systems that include a return channel was recently specified by ETSI in [4].

One of the main hurdles in making these systems competitive from a user standpoint is that they do not provide very good performance for TCP-based applications such as HTTP. The performance degradation of TCP over point-to-point GEO satellite links is well understood by now (for a more detailed discussion on this topic refer to [7], [8]). To briefly summarize, significant bandwidth asymmetry, large amount of acknowledgment traffic generated by TCP over the return channel, large round trip delays (about 540 ms for GEO satellite links), small default window sizes of most client-server implementations, and unfairness of TCP toward connections with larger round trip delays are some of the reasons why TCP does not perform well over point-to-point satellite links. Several mitigations have been proposed to alleviate these problems. In particular splitting a TCP connection into three different connections by having two proxies, one at each end of the point-to-point satellite link, is known to provide considerable performance gains.

However most of the work that has been done in this area so far has considered only a point-to-point satellite link, i.e., a link where there is no sharing of uplink bandwidth. On the other hand, in several existing and proposed satellite systems, there is a single hub connected to the Internet with a fast connection, and a large number of users connected to this hub through a multi-access satellite link (a more detailed discussion on this scenario follows in section III). Uplink bandwidth can be shared in several possible ways, using slotted Aloha (very low throughput), uniform static allocation to all the terminals (poor utilization), and a pure Bandwidth on Demand (BoD) wherein before transmitting any packet, the terminal has to request permission from the hub (large delays and so low throughput). None of these solutions by itself is good for TCP based applications and hence there is a need to design a smart MAC protocol. As can be seen through our studies, this MAC layer has considerable influence on the higher layer protocols even with splitting. This was our motivation behind studying TCP splitting in a multi-access GEO satellite system. Our work can be easily extended to the case of on-board processor satellite systems with some modifications. Most of the work done so far has focused on per connection performance, however we take an approach in which we study the performance of the entire satellite system.

This paper has been organized in the following way. In section II, we discuss the previous relevant work in this area. In section III, we discuss the network scenario that we are considering. In section IV, we discuss the salient features of Satellite Transport Protocol or STP (for a more detailed discussion on STP, refer to [13]). Then in section V, we briefly discuss the DVB-RCS [4] standard and in section VI we propose a new MAC protocol that is consistent with this standard . In section VII, we present some simulation results. In section VIII we talk of possible future work and our conclusions.

II. Previous work

TCP performance over a satellite link can be improved by adopting either one of the following two approaches; making changes at the end client and server TCP stacks (most of the mitigations proposed in [5] and [6]) or putting transport layer proxies at each end of a satellite link which have been tuned to perform well over satellite links. The first option is not feasible since modifying TCP stacks of all the clients and servers is not practical. The second option has been explored by a lot of researchers with various implementations of proxies. [9] proposes splitting and an optimized wireless link layer with selective ARQ. In [12] the authors propose a TCP splitting solution with link level retransmissions to tackle packet losses due to errors over the satellite link. In both these solutions, the authors propose using TCP as the transport protocol over the satellite link, but with certain modifications.

Then there is the work on the Satellite Transport Protocol (STP) [13], in which the authors have proposed a new transport protocol for the split connection segment over the satellite link. The STP has several advantages over other TCP based solutions such as very large window sizes, low return channel usage, selective acknowledgments with a facility for exclusive negative acknowledgments, etc. However the protocol was mainly designed for point-to-point satellite links. As a result its performance may not be equally good, if the underlying MAC protocol adds variable delays in packet transmission. We propose a new MAC protocol for uplink bandwidth sharing which is fine-tuned to work together with STP.

III. Network Scenario

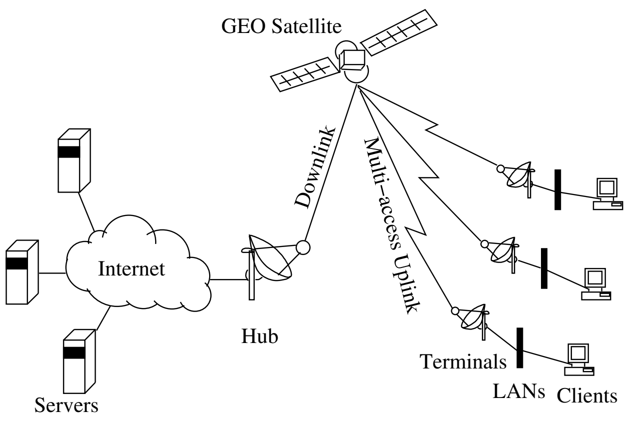

Fig. 1. Network Scenario: Broadcast Downlink and Multiple-access Uplink

Fig. 1 illustrates the geostationary bent-pipe satellite system that we consider. In such a system, there is a central hub which is connected to the Internet by a fast connection. Terminals connect the users' equipments (i.e., PCs) to the hub through the satellite system. The hub uses the downlink to send packets to all the terminals while the terminals share the uplink channel to send packets to the hub. There is considerable asymmetry between the downlink and the uplink (sometimes referred to as forward and return channels respectively), since the downlink is broadcast with high capacity while the uplink is multi-access (MF-TDMA) and has a much lower capacity. This bandwidth asymmetry should suit the traditional client-server applications since most user terminals are connected to clients. Besides, a large portion of the data consists of Web traffic. As a result, most of the data is expected to flow from the hub to the terminals. In particular for HTTP traffic, the uplink data consists only of acknowledgments and small HTTP GET requests to retrieve data from the server (excluding the initial connection establishment SYN packets). However, even if the data flow in itself is asymmetric, there could be a large number of terminals connected to the hub, and not all the terminals are active at the same time. Even among the terminals which are active, the intensity of traffic to and from each terminal may vary considerably.

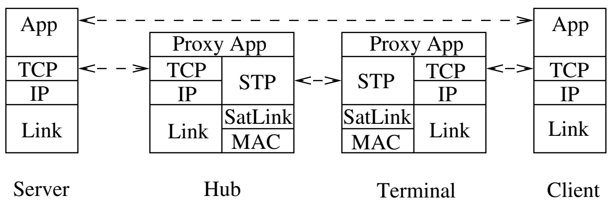

Fig. 2. Single TCP connection split into three connections, TCP-STP-TCP

The MAC protocol design should consider all these factors and co-ordinate the sharing of the uplink bandwidth between the terminals. Besides, for HTTP like applications, the key design objective is to minimize the response time of retrieving an object once a request has been sent. We consider the TCP splitting solution where we use STP as the transport protocol over the satellite segment of the split connection as illustrated in Fig. 2. The proxy application does the job of transferring data from STP's buffer to TCP's buffer (or TCP's buffer to STP's buffer) depending on the amount of free buffer space available in the TCP's buffer (STP's buffer). Thus, it is the proxy application that is responsible for the flow control co-ordination of the three segments.

IV. Satellite Transport Protocol (STP)

STP is a transport protocol that was designed to perform well over a satellite link. It is a modification of the original SSCOP protocol [14]. The STP sender and receiver use buffer sizes that are of the order of the bandwidth-delay product of the link. Satellite links are characterized by a large bandwidth-delay product due to very high round trip times and therefore most common implementations of clients and servers who use 4KB to 8KB of window sizes do not operate well over the satellite links. STP solves this problem by having a window size as large as 240KB. STP also uses selective negative acknowledgments to recover from packet losses. The acknowledgment scheme of STP is interrupt driven and not packet driven like TCP wherein the sender periodically polls the receiver and requests for acknowledgments. The polling period could be as high as one RTT, which means that irrespective of how fast data is sent to the receiver, only one acknowledgment needs to be sent during one RTT. In case of TCP, if n TCP segments are sent during one RTT, as many as $n/2$ acknowledgments need to be sent. This results in a much lower return channel usage in case of STP. Polling enables the transmitter to flush out those packets which have been successfully received by the receiver and retransmit only the lost packets. STP uses a TCP-like congestion control scheme, however there are no retransmit timers associated with each segment (since there is periodic polling). Slow start is entered only once when the connection is first established. All these design features make STP an ideal choice for a transport protocol over the satellite link. We made a small change to the STP implementation from [13] wherein the sender (i.e., the hub) polls the receiver only if there is unacknowledged data in the pipe. This reduces the control traffic for persistent HTTP connections which often have long silent periods. The acknowledgment packets from all the STP receivers in a terminal were also aggregated into a single packet for ease of bandwidth allocation at the hub and also to minimize the bandwidth overhead. At the hub, the information in this packet was then demultiplexed and passed on to the STP senders.

We also observed a few subtle issues that arise if we try to use STP as it is over a multi-access link. We know that the acknowledgment and the control traffic generated by an STP receiver over the return channel is very little as compared to the TCP ACK traffic. This also implies that very few acknowledgment packets are generated by the receiver and each of these acknowledgment packets is very important from the point of view of flow control, congestion control and error recovery. So any delay experienced by these control packets seriously degrades the connection throughput. In presence of a MAC layer over a high delay satellite link, packets might have to face large and variable media access delays.

Through our studies we found out that these delays were not a big problem for HTTP like applications where more than throughput, it is the response time that matters. With the large buffer sizes that STP uses, even if the acknowledgments get delayed by small amount, the performance still does not degrade. However for bulk transfer applications such as FTP, the MAC layer latencies can cause a reduction in throughput. This however is not the topic of this paper. As we will demonstrate, by designing a smart MAC, even STP performance for HTTP like applications can be improved. The MAC protocol that we propose is fully compliant with the DVB-RCS ([4]) standard, which we briefly discuss in the next section.

V. DVB-RCS MAC

The DVB-RCS standard, [4] specifies a MAC layer in which the hub controls the sharing of the common satellite uplink by the terminals. The uplink is MF-TDMA which basically means that there are several TDMA channels, each being subdivided into frames. Each frame consists of time slots of fixed length. This fixed length could either correspond to one ATM cell of 53 byte length (48 bytes for payload), several ATM cells, or MPEG-2 packets of 188 byte length (184 bytes for payload). When a terminal wishes to transmit data over the uplink, it first explicitly requests for the correct number of time slots from the hub. The hub receives the time slot requests from all the terminals and periodically runs a scheduling algorithm for allocating the available time slots and creates an allocation table called the Burst Time Plan (BTP). The hub then broadcasts this BTP to all the terminals. The terminals look at the received BTP and then transmit their data during the alloted time slots. Besides this request-allocation procedure, the hub could also allocate a fixed number of time slots to certain terminals on a periodic basis, thereby guaranteeing a certain minimum uplink bandwidth to those terminals.

VI. Proposed MAC Protocol

Since the uplink traffic mainly consists of acknowledgments and bursty short data packets, allocating a static rate to a terminal over the uplink leads to a wastage of uplink time slots. Besides, in presence of a large number of terminals with varying activities, the static allocation of bandwidth means that there are terminals that need more bandwidth but cannot have any, while there are terminals that do not need all the bandwidth alloted to them. Hence a pure static allocation of bandwidth is not a good solution. On the other hand having the terminal request time slot for each and every packet would mean that the all important HTTP GET requests get delayed leading to a higher response time.

We propose a bandwidth allocation algorithm which is a combination of Bandwidth on Demand and dynamic allocation of free bandwidth. In this scheme, the terminals look at their current interface queue length and the number of time slots allocated to them in the current BTP (Burst Time Plan), and then request the required number of time slots from the hub. These requests are periodically piggybacked with data frames. The hub does a fair allocation of available bandwidth to satisfy the requests. If there are any free time slots left after this allocation, they are again distributed fairly among the terminals. By "fair", we mean fairness in terms of the number of connections. The MAC layer gets the information about the number of active connections between the hub and a given terminal from the underlying transport layer. The weight assigned to each terminal is proportional to the number of STP connections in it. Bandwidth on Demand ensures that our MAC protocol is responsive to the occasional traffic bursts at the terminals, while the free time slot allocation based on the number of connections ensures fair sharing of the available bandwidth.

VII. Simulation Results and Discussions

We used *ns-2* (version 2.1b8a) for our simulations. The simulator has some built-in support for creating various satellite network topologies. With the original simulator, it is possible to create a satellite network with a GEO satellite node in the sky and several terminals on the ground. Each of these nodes have an interface consisting of a physical layer, a MAC layer and a satellite link layer. However the only MAC layers supported were point-to-point link and slotted ALOHA. Besides the routing modules did not support a topology consisting of satellite as well as terrestrial nodes. We added new routing modules to the simulator so that the satellite nodes and the terrestrial nodes could co-exist in the same topology. This enabled us to simulate a truly heterogeneous network consisting of terrestrial client nodes connected to the satellite hub through the terminals over the satellite link.

We added new interface objects so as to simulate a channel that has a broadcast downlink and a multi-access uplink. We added a new MAC layer for simulating MF-TDMA over the uplink with Bandwidth on Demand (BoD). In this design, the uplink is time slotted into slots of 53 byte length each to match the DVB-RCS specifications. The terminals located at different geographical locations were synchronized through the periodic beacon signals sent by the hub, and this ensured that in spite of different propagation delays, all the terminals had a common notion of the time slot allocation. The terminals can also request time slots from the hub by piggybacking their requests on data frames.

The original implementation of TCP and STP in *ns* did not have any support for flow control since applications were assumed to consume the data passed on to them by the transport layer instantaneously. The proxy application however consumes data from one of the transport agents and passes it on to the other if there is free space in the receiving agent's buffer. This means that the application consumes data at a rate which depends on the state of the two underlying transport agents. Hence we implemented flow control in TCP as well as STP to simulate the correct overall flow control behavior. Modifications were made to the STP implementation obtained from [15] so as to make it fully duplex. We used send and receive buffers of size 8KB for standard TCP, and 64KB for STP. The STP sender polls the receiver once every two round trip times.

We used these modified duplex TCP and STP agents together with the web caching support in *ns* to simulate a fully duplex HTTP agents. The HTTP agent at the client sends HTTP GET messages for retrieving objects from the server and the server tries to send the data as fast as the network allows it to. Each agent simulates a persistent HTTP/1.1 connection in which GET requests are pipelined back to back over the same TCP connection. Each client agent generates requests with a Poisson point process model once every 2 seconds. The client always requests objects of fixed size (4KB). The simulator scripts that were developed in this work could be found at [16]

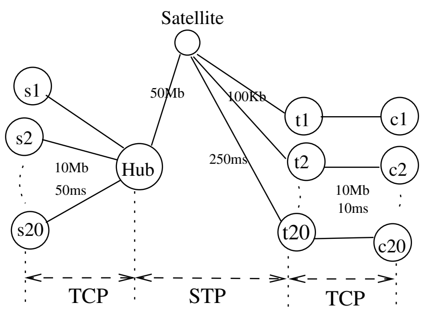

Fig. 3. Simulation Scenario: 20 terminals connected to the hub.

Fig. 3 illustrates the network topology that we simulated. It consists of 20 terrestrial user PCs connected to 20 satellite terminals through a 10Mbps link. These satellite terminals were connected to a central hub through a satellite link. This satellite link is characterized by an uplink bandwidth of 100Kbps and a downlink bandwidth of 50Mbps. We chose these settings because we were interested in studying the effects of uplink being the bottleneck and so we over-provisioned the downlink so as to remove any effects of congestion over the downlink. The 100Kbps bandwidth of the uplink was to be shared by the 20 terminals by using a MAC protocol which was controlled by the hub. The uplink is time slotted into slots of 53 byte length each of which 48 bytes are for data (DVB-RCS standard [4]). This effectively leaves a useful bandwidth of about 91Kbps. The hub is connected to 20 servers through 10Mbps links. Again we over-provision the bandwidth on these links and have separate server for each client with a separate connection since we do not want the effects of congestion on the terrestrial links to impact our study of the multi-access uplink. Typical satellite networks have a small uplink capacity which is to be shared by the terminals and the downlink is usually a high capacity channel. We are trying to simulate such scenarios. In order to simulate the heterogeneity of the user population, we attached 5 HTTP agents to 10 user nodes and 1 HTTP agent to the rest of the 10 user nodes each (a total of 60 HTTP connections). The simulation period was chosen to be 80 seconds.

We simulated three different scenarios:

A. Scenario 1: TCP end-to-end with static sharing of uplink and no BoD

This is the worst possible scenario in which the uplink bandwidth is equally divided between all the terminals and there is no Bandwidth on Demand. If there is extra traffic at one terminal and no traffic at another, there cannot be any redistribution of bandwidth. We also tried to simulate TCP with BoD, but the large BoD delays caused TCP to timeout on most occasions.

B. Scenario 2: TCP splitting with STP, BoD and uniform sharing of remaining capacity

In this scenario, we split an end to end TCP connection into a TCP connection between the client and the terminal, an STP connection between the terminal and the hub and a TCP connection between the hub and the server. During connection establishment, a proxy does not reply to the client until it receives a reply from the server. This prevents the proxy from responding to a client without confirming the existence of the server. The terminals request time slots from the hub for transmitting data over the uplink every 250ms. The hub tries to satisfy all the bandwidth requests and then distributes the remaining bandwidth equally among all the terminals. The hub broadcasts the BTP once every 250ms.

C. Scenario 3: TCP splitting with STP, BoD and non-uniform sharing of remaining capacity

This scenario is similar to the one described above with a slight modification to the BoD algorithm. In this case, the hub allocates the free time slots among the terminals in proportion of the number of connections in that terminal. This ensures that the terminals who currently have a high traffic load get a higher share of the available free bandwidth which in turn reduces their BoD delays. The MAC layer at the hub gets information about the current number of connections to all the terminals from the underlying STP agents.

From the above simulation results in Table I, the following observations can be made:

- Using TCP as it is over a satellite link with static bandwidth allocation over a low bandwidth channel leads to extremely poor HTTP performance with large delays and very low throughput. It also uses a lot of return channel bandwidth due to the frequent acknowledgments that are sent from the clients to the server. This is not surprising

	Scenario 1	Scenario 2	Scenario 3
Total no. of HTTP objects retrieved	1106	1628	1824
Mean Response time (seconds)	2.25	0.87	0.83
Maximum Response time (seconds)	13.14	8.42	7.49
Downlink bandwidth usage (Kbps)	778	1155	1291
Uplink bandwidth	44.77	23.91	26.27

TABLE I
COMPARISON OF HTTP PERFORMANCE FOR THREE DIFFERENT SCENARIOS.

since the performance degradation of TCP over satellite links is already well known.

- TCP splitting with STP and BoD gives an excellent performance as compared to end to end TCP. Clients can retrieve about 47% more objects and sustain half the delay as the previous case. STP works on periodic polling of the receiver by the sender and this polling period can be very large since the large STP buffers ensure that the data flow is not slowed down if acknowledgments are delayed. Hence even with BoD, STP can function efficiently. This is reflected in the small response time in Table I. However we could do better than this if the bandwidth allocation explicitly takes into account the activity of each terminal. This happens in scenario 3.
- In this scenario, the MAC layer at the hub knows the number of connections in each terminal and this helps it anticipate the amount of traffic at each terminal. So after allocating all the requested BoD time slots, if there are any more time slots left, the hub allocates them to the terminal in proportion of the number of STP connections in that terminal. This results in better response time and higher throughput since terminals get time slots based on their activity. There is a noticeable increase of 12% in the number of HTTP objects retrieved as compared to scenario 2 and a whopping 65% increase as compared to scenario 1. Also note that both scenario 2 and 3 have almost half the bandwidth usage of the return channel. This can be attributed to the STP. Thus STP ensures low return channel usage while dynamic rate allocation with BoD ensures small response times for HTTP access.

VIII. CONCLUSIONS

In this paper we studied the effects of bandwidth sharing in a multi-access uplink on HTTP page transfer with TCP splitting and STP. We observed that our BoD scheme from scenario 3 integrates well with STP for HTTP like applications. We also observed that by making the BoD aware of the number of connections in each terminal, it is possible to get even better throughput and smaller response times. We also developed the *ns* simulator to support a whole range of features for studying satellite networking, proxies and TCP flow control.

In the future we would like to investigate how the bulk transfer of data with TCP splitting and STP is affected by the multi-access nature of the uplink. As was mentioned earlier, the large varying BoD delays are likely to have a more profound impact on the bulk data transfer than the HTTP transfer since in bulk transfer, the flow control co-ordination of STP and TCP gets directly affected by the delays faced by the STP control traffic. We are currently working on developing a MAC protocol which decouples data and acknowledgment traffic of STP to ensure prompt delivery of acknowledgments so that the BoD delays do not slow down the co-ordinated flow control mechanism. We are also doing an analysis of our protocol to quantitatively evaluate its performance gains.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. 0087266.

REFERENCES

[1] J. Postel, "Transmission Control Protocol - protocol specification," informational, Internet RFC 793, Sept. 1981.
[2] W. Stevens, *TCP/IP Illustrated Volume 1.* Addison-Wesley, 2000.
[3] The *ns* Manual (for version 2.1b8a) `http://www.isi.edu/nsnam/ns/ns-documentation.html`
[4] ETSI EN 301 790 V1.2.2 (2000-12), Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems, *European Standard (Telecommunications Series).*
[5] M. Allman, *et al.*, "Enhancing TCP Over Satellite Channels using Standard Mechanisms," informational, Internet RFC 2488, Jan. 1999.
[6] M. Allman, *et al.* "Ongoing TCP Research Related to Satellites," informational, Internet RFC 2760, Oct. 1999.
[7] H. Balakrishnan, V. Padmanabhan, and R. Katz, "The Effects of Asymmetry on TCP Performance," in *3rd ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, Budapest, Hungary, Sept. 1997.
[8] H. Kruse, M. Allman, J. Grinner, D. Tran, "HTTP Page Transfer Rates over Geo-Stationary Satellite Links," *Proceedings of the Sixth International Conference on Telecommunications,* Nashville, TN, Mar. 1998.
[9] J. Scott Stadler and J. Gelman, "Performance Enhancement of TCP/IP on a Satellite Channel," MIT Lincoln Laboratory, *IEEE Communications,* Jan. 1998.
[10] G. Acar and C. Rosenberg, "Weighted Fair Bandwidth on Demand (WFBoD) for Geo-Stationary Networks with On-Board Processing," *Special Issue on Broadband Satellite Systems: A Network Perspective, Computer Networks,* Vol. 39, Issue 1, May 2002.
[11] G. Acar and C. Rosenberg, "Algorithms for computing Resource Requests in Bandwidth on Demand in a Satellite Access Unit," 5th Ka-band Utilization Conference, Taormina, Italy, Oct. 1999.
[12] M. Karaliopoulos, R. Tafazolli, B. Evans, "Enhancing TCP's performance over GEO satellite links with splitting connections and link level retransmissions," *Proceedings of the 19th AIAA International Communications Satellite Systems Conference (ICSSC '01),* Toulouse, France, Apr. 2001.
[13] T. Henderson and R. Katz, "Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks," *Second Workshop on Satellite-Based Information Systems, (WOSBIS-97),* Budapest, Hungary, Oct. 1997.
[14] T. Henderson, "Design Principles and Performance Analysis of SSCOP: A New ATM Adaptation Layer Protocol," *ACM Sigcomm: Computer Communications Review* Apr. 1995.
[15] T. Henderson, Satellite Transport Protocol, simulator scripts for *ns* `http://www.tomh.org/software/stp_ns.tar`
[16] V. Mhatre, TCP over Multi-access Satellite Networks, simulator scripts for *ns* `http://icdweb.cc.purdue.edu/~mhatre/ma/`