Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks

Thomas R. Henderson* and Randy H. Katz
University of California at Berkeley
{tomh,randy}@cs.berkeley.edu

*also with Hughes Research Laboratories, Malibu, CA

Abstract-- **We describe the adaptation of an ATM-based protocol, the Service Specific Connection Oriented Protocol (SSCOP), for use as a transport protocol in a datagram-based satellite data network. This protocol operates with high throughput and efficiency over links with a large bandwidth-delay product, including those with a high bit error or packet loss rate. We discuss the modifications to SSCOP necessary for it to operate over an IP-based satellite network. We then compare the performance of the modified protocol (the Satellite Transport Protocol, or STP) with that of conventional TCP and TCP with selective acknowledgments (TCP-Sack). We find that STP performs comparably to TCP-Sack in terms of forward throughput over simulated satellite channels. However, STP uses significantly less bandwidth on the return path, and is relatively insensitive to variations in round trip delay, making it an attractive candidate for asymmetric networks. Finally, STP exhibits very good performance in a rate controlled environment, even when the bit error ratio is severely degraded.**

1. Introduction

One of the main challenges of data communications over satellite networks is the efficient transport of data over connections with a large bandwidth-round trip delay product (BW*RTD). In addition to dealing with large latencies, transport protocols must also contend with asymmetries in bandwidth, jitter, and error performance in the forward and reverse path. The most popular unicast transport protocol in use today, the Transmission Control Protocol (TCP) [Pos81], is known to suffer from performance degradation in satellite and asymmetric environments [Dur96, Bal97a]. Some deficiencies, such as the small protocol modulus and the error recovery mechanism, have been remedied with the publishing of new protocol options such as "TCP Extensions for High Performance" [JBB92] and "TCP Selective Acknowledgment Options" [MMFR96] (although it is unclear to what extent these options will become standard practice in the future). Other problems with TCP, such as the bias inherent in the congestion avoidance algorithm [Hen97], an inability to resize the receiver resequencing buffer based on the observed round trip time (RTT), performance problems with widely varying round trip times, and the breakdown of self-clocking in an asymmetric environment [Bal97a], have not been corrected in the standards. Since the latencies and asymmetries in satellite networking are not likely to disappear in the near future, it is important to consider what types of improvements are possible.

There are four basic strategies for improving TCP performance over satellite networks or wireless networks in general. First, one can tinker with the implementation of TCP, through the use of either standard or non-standard options or protocol changes. The main disadvantage of this approach is that the enhanced protocol must remain backward compatible with current TCP, since the changes must be incrementally deployed in the network. Some proposed enhancements to TCP require that both hosts involved in the connection implement the changes. For example, the ACK suppression strategy for asymmetric connections, described in [Bal97a], requires that the sending TCP increase its window based on the amount of data acknowledged, and not on the physical number of ACKs. If the sender does not implement the change, however, the window growth can be seriously retarded by ACK suppression. Furthermore, protocol enhancements which do not require changes to the peer's implementation (such as a more aggressive window growth algorithm, examined in [Hen97]), often offer only modest benefits and do not correct fundamental performance limitations of the protocol. Finally, there is a wide diversity of implementations in existence today [Pax97], making it difficult to predict the behavior of the implementation at the peer host.

A second approach for improving TCP performance is known as "*striping*." This technique, known to satellite operators for some time, has more recently been investigated in the context of TCP by Allman [All96] and

Balakrishnan [Bal97b]. The approach involves opening multiple simultaneous connections, coordinated by a session manager, in order to improve the throughput over what would be obtained by using a single connection. The main drawback to this approach is that it requires cooperation by the peer host (in maintaining a session manager), and the underlying performance characteristics of the individual TCP connections is not improved.[1]

The third common approach is known as "*spoofing*." In this case, an agent in the network works on behalf of a host system by, for example, altering the ACK stream in order to open up the congestion window more rapidly [Zha97]. While not requiring changes to the end systems, it does require the presence of an agent in the network (such as at the gateway of the satellite system), cannot be used in instances in which the IP payload (i.e., TCP header) is encrypted, and does violate the end-to-end semantics of the transport layer service model. A related approach proposed for wireless systems, known as "*snooping*," is implemented similarly but does not violate the end-to-end semantics of TCP [Bal95].

The final strategy is known as "*splitting*." In this case, the end-to-end TCP connection is actually broken in two by an agent within the network, and this agent implements either a modified or a new protocol for transport over the difficult network segment ([Bak95] and [Dur96]). Unlike spoofing, the end hosts are aware of the presence of the intermediate agent in the network. An example application would be if a satellite system were to instruct its users to redirect HTTP requests through a WWW proxy.

In this paper, we describe a transport protocol that is appropriate for use either for providing the satellite segment of a split TCP connection or for providing a reliable channel for control traffic within the satellite network. It is suitable for both connection-oriented and connectionless satellite networks. This protocol is based heavily on one that has been implemented in ATM networks, called the Service Specific Connection Oriented Protocol (SSCOP). SSCOP was targeted for large BW*RTD networks, and hence performs quite well over satellite channels [Chi94]. As it was specified, SSCOP relies on in-sequence data delivery by the lower protocol layers. In this paper we describe the necessary modifications and additions to make SSCOP work in a connectionless network environment such as IP. The basic features of the protocol, which we refer to as the Satellite Transport Protocol (STP), have been implemented in the UCB/LBNL Network Simulator *ns,* and we discuss some early results. In particular, we show that STP outperforms TCP in networks with a high BW*RTD product, and especially over connections which have constrained reverse channel bandwidths.

[1] This technique is used by popular WWW browsers today, but may be phased out if persistent TCP connections are used (such as in HTTP 1.1).

2. History and related work

SSCOP is the result of an international standardization effort from the period 1990-1994 [ANS94, ITU94]. When originally conceived, SSCOP was intended to provide both a reliable link layer for ATM signaling and a reliable transport layer for user data over ATM networks. In ATM, the core mechanism of SSCOP can be augmented by Service Specific Coordination Functions (SSCFs), to provide different types of service interfaces to the higher layers, and four such SSCFs have been standardized by the ITU. These SSFCs can be thought of as providing a different API to the user.[2] SSCOP is currently being used for ATM signaling at both the UNI and the NNI. It is not being used for user data transfer, however, since ATM is generally being used as a subnet technology for IP networking. Since the current specification of SSCOP relies on guaranteed in-sequence data delivery, it is not appropriate to use it in a datagram network.

The following were the fundamental design goals of SSCOP. The two main design goals were *optimization for high speed operation* (i.e., low processing overhead), and *efficient operation in networks with large BW*RTD*. More specifically [Hen95]:

- the protocol modulus is large;[3] the sequence number is 24 bits and data is numbered per-packet rather than per-byte;
- the protocol is 32 bit aligned, and the control information is trailer oriented in order to reduce hardware bus cycles in unnecessary memory-to-memory copies (this design goal presumed that SSCOP would eventually be implemented in hardware);

[2] The HDLC class of data link protocols was constructed in a similar manner, with a base level specification forming a "toolbox" from which other protocols, such as ISDN LAPD, can be defined.

[3] It was once noted at a standardization meeting that the modulus should allow for a 10 Gb/s connection to traverse two geosynchronous satellite hops.

- error recovery is based only on selective retransmission;
- control and information flow is separated, reducing per-packet overhead;
- protocol logic is decoupled from timers, reducing the dependence of the implementation on timers and allowing for variations in round trip delay; and
- the transmitter and receiver can be decoupled and do not share state variables, allowing for partitioned implementation.

There have been a number of new transport protocols proposed and implemented over the past decade (e.g., NETBLT, XTP, Datakit, Delta-t, and SNR) [Doe90]. Although the development of SSCOP was influenced by the contributions of a number of organizations, SSCOP bears the closest resemblance to a protocol known as SNR (after its inventors Sabnani, Netravali, and Roome)[Net90]; the authors of SNR were also involved in the early development of SSCOP. The protocol described herein differs from SNR in that it is specifically targeted for use in IP-based or similar networks, and incorporates some different features (for example, SSCOP's USTAT message, described below, has no SNR analog). Nevertheless, it should be emphasized that the basic concept of SNR, the separation of control and data flow coupled with periodic exchanges of protocol state, is carried over into STP.

Finally, it is important to mention that researchers interested in modifying TCP for the space environment have independently developed protocol features that resemble those in SSCOP. Most notably, explicit negative acknowledgments and features for handling asymmetric channels, both fundamental features of SSCOP, are proposed for the Space Communications Protocol Standards—Transport Protocol (SCPS-TP) [Dur96]. We have found one additional feature in SCPS-TP, discrimination between lost and errored segments in the flow control algorithm, useful for STP as well.

3. Basic operation of STP

STP and SSCOP are examples of synchronous bit oriented protocols. Such protocols are responsible for flow control, sequencing and accounting of data, and response to error detection [Bla93]. The main function of the protocol is to reliably transfer data to its peer in variable length packets. The transmitter sends packets to the receiver, storing the packets for potential retransmission until the receiver has acknowledged them. STP's automatic repeat request (ARQ) mechanism uses selective acknowledgments only. Packets are numbered sequentially, and the transmitter retransmits only those specific packets that have been explicitly requested by the receiver (i.e., there are no "timeouts") [Hen95].

3.1 Protocol operation

STP has four basic packet types for data transfer (we are ignoring, for now, the additional packet types needed for connection setup and release). The Sequenced Data (SD) packet is simply a variable length segment of user data, together with a 24 bit sequence number and a checksum. SD packets which have not yet been acknowledged are stored in a buffer, along with a timestamp indicating the last time that they were sent to the receiver. No control data is included in the SD packets; instead, the transmitter and receiver exchange "POLL" and "STAT(us)" messages. Periodically, the transmitter sends a POLL packet to the receiver; this POLL packet contains a timestamp, and the sequence number of the next in-sequence SD packet to be sent. The receiver responds to the POLL by issuing a STAT message which echoes the timestamp, includes the current window value of the receiver and the sequence number of the highest in-sequence packet to have been successfully received, and contains a listing of all gaps in the sequence number space spanning from the highest in-sequence packet received to the sequence number reported by the POLL packet. The STAT message is similar in concept to a TCP selective acknowledgment (SACK), except that the STAT message reports the entire state of the receiver buffer (rather than the most recent three gaps in a SACK), and the STP receiver may not renege on previously acknowledged data (which is permitted for TCP with selective acknowledgments). Large STAT messages can be segmented, and each STAT message is a complete report of the state of the receiver; these features allow for resiliency in the event of lost POLL or STAT packets. Finally, the receiver can independently report on missing packets via a "USTAT" (unsolicited STAT) packet. This packet allows the receiver to quickly stimulate a retransmission, and in so doing permits the POLL and STAT exchange to be conducted less frequently. The USTAT is an explicit negative acknowledgment requesting immediate retransmission.

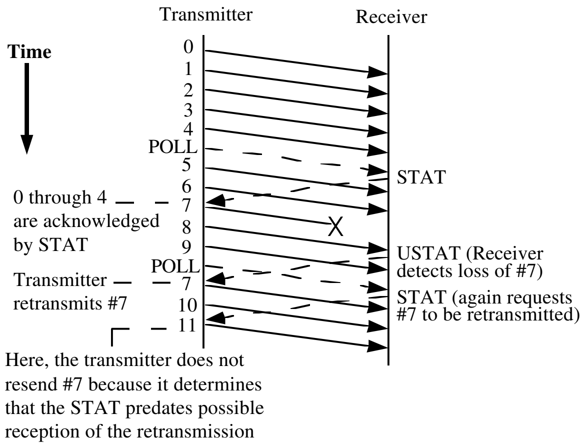

Figure 1: Example of SSCOP operation.

The basic operation of STP (and of SSCOP) can best be understood by considering the example illustrated in Figure 1 (from [Hen95]). This example only illustrates one direction of data transfer, and assumes that sequence integrity of transmissions is preserved (in the next section, we describe how to eliminate this assumption). In the example, the transmitter sends a series of consecutively numbered packets. After packet #4 is sent, a POLL packet is sent (either due to the retransmission of a POLL timer or a threshold on the number of new packets sent). The POLL tells the receiver that the next message to be sent is #5, so the receiver knows that it should have received packets 0 through 4. In this case, since they have been received, the receiver returns a STAT packet acknowledging all data up to and including packet #4. After sending the POLL, the transmitter continues with packets 5 through 9; however, packet #7 is lost. The receiver detects this loss upon receipt of packet #8 and immediately requests retransmission of #7 with a USTAT packet. Before this USTAT is received at the transmitter, the transmitter again sends a POLL packet. Upon reception of the USTAT, the transmitter immediately resends #7, and continues on with #10 and #11. Before the retransmission is received, the receiver responds to the POLL by again requesting #7. However, the timestamp in the STAT packet allows the transmitter to determine that its retransmission has not yet had an opportunity to reach the receiver, thereby avoiding an unnecessary retransmission. If #7 were instead lost again, the next STAT message would have stimulated a second retransmission. More examples of basic SSCOP operation can be found in an appendix of the protocol specification [ITU94].

3.2 Packet formats

The STP packet formats are illustrated in Figure 2. They illustrate the requirements for running over IP; modifications may be necessary if the underlying network protocol is not IP. All packets are 32 bit aligned, but they are not trailer oriented as in SSCOP (since IP is header oriented and memory-to-memory copies can be avoided by leaving room for packet headers in memory buffers). We first describe header fields common to all packets. As in TCP, each packet has a source and destination port ID, which serve to identify the particular transport connection, in the first 4 bytes. The next byte contains a 4 bit "packet type" field in the highest four bits. This byte is followed by a 24 bit sequence number. For SD packets, it is assumed that the user data is byte aligned. If the data is not 32 bit aligned, up to 3 bytes of padding are added, and the length of this PAD field is coded in the 2 bits following the packet type. The sequence number field in the SD packet contains the sequence number of the packet. For the POLL packet, the sequence number field reports the sequence number of the next in-sequence packet to be transmitted. For the STAT and USTAT packets, the sequence number field indicates the highest in-sequence packet received. The third 32 bit word of the packet header contains a checksum which covers the entire STP packet. The selection of a checksum depends on the error protection of the underlying layers. We have left room for a 32 bit cyclic redundancy check (CRC), although an implementation can use a shorter checksum (such as the standard 16 bit IP checksum) if a strong CRC is used at the link layer. Any unused bits in the headers are *reserved*; they are coded as "0" by the transmitter and ignored by the receiver.

In addition to these common header fields, POLL, STAT, and USTAT packets contain additional fields. The POLL packet encodes a timestamp value corresponding to the time at which the POLL is sent. This timestamp is echoed in the STAT packet. The STAT packet also contains a value for the receiver window (measured in packets). In addition, if there are any gaps in the receiver's buffer, they are reported as a number of list elements. A pair of list elements denotes the starting and ending sequence numbers of a gap. As a performance enhancement for error-prone links, if the receiver guesses that the gap was due to a checksum failure rather than

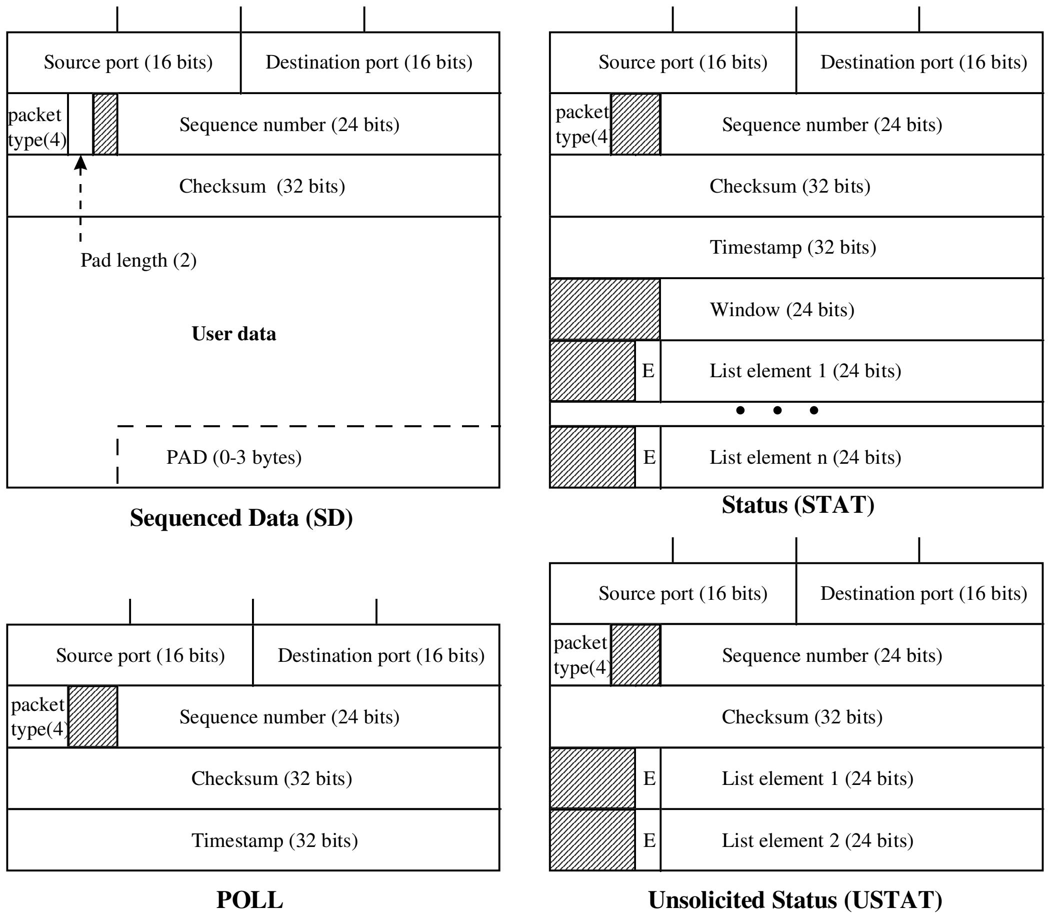

Figure 2: Four basic STP packet types.

the absence of the packet, the "E(rror)" bit is set for that entry. Large STAT packets can be segmented if they exceed some configured packet size. Finally, the USTAT packet is a simplified version of the STAT, reporting only the most recently observed gap in the sequence buffer.

4. New STP features

4.1 Modifications to SSCOP for operation over IP

The fundamental change that must be made to SSCOP in order for it to work over IP is a relaxation on the restriction of out-of-order packets. Since there are no unnecessary retransmissions possible in the connection-oriented protocol logic, reception of a duplicate packet is considered a protocol error. This is easily remedied by allowing for silent discard of redundant data packets, which may occur in a datagram network.

A more interesting modification must be made to the retransmission procedure itself. In particular, the presence of a sequence gap does not necessarily indicate that a packet has been lost. Retransmissions should be delayed until a point in time when it is unlikely that the packet has merely been mis-sequenced by the network. The burden of waiting could be placed either at the transmitter or the receiver. Since it is easier to maintain accurate round trip delay estimates at the transmitter (based on timestamp exchange in the POLL/STAT exchange), we favor the following approach: the STAT sent by the receiver reports exactly the current state of the receiver, and the transmitter decides whether or not to retransmit missing packets based on a comparison of the timestamp of the STAT and the timestamp stored and associated with the missing packets. TCP has implemented a technique for estimating the mean deviation of the round trip time of a connection (the mean deviation is an easier computation than the standard deviation) [Jac88]. The following pseudocode is suggested for deciding whether or not a packet reported as missing by the receiver is eligible for retransmission:

```
if (STAT_ts - stored_ts > k*mdev)
    retransmit;
```

where *k* is a constant such as 2, *ts* stands for timestamp, and *mdev* is the current value of the mean deviation in round trip delay. The value of *k* is subject to tuning based on the likelihood of misordered packets; a suboptimal value, however, is not catastrophic since it only results in either a longer delay in retransmitting or more unnecessary retransmissions.

Recall that another feature of SSCOP was the transmission of a USTAT message based on the immediate detection of a sequence number gap. Obviously, such a feature should not strictly be used in a network that reorders packets. However, we would like to retain the mechanism because it allows more timely recovery of lost packets without the need for a frequent POLL/STAT exchange. The solution is to delay the sending of USTATs until the sequence number has exceeded the missing packet (or right edge of a gap of packets) by n packets. Setting the parameter n to 3 makes the procedure analogous to the "fast retransmit" facility in TCP. Just as TCP's fast retransmit only occurs upon the first retransmission of a packet, the USTAT is only sent upon the first detection of a sequence gap; subsequent retransmissions are based on STAT reports.

The SSCOP protocol specification has no flow control mechanism besides a sliding window capability; for the ATM signaling links for which it was initially designed, a sophisticated scheme was not needed since the signaling links are high priority and of predictable traffic load. For data transfer in a distributed packet-switched network, some mechanism is needed to adapt to changing network conditions. The TCP flow control mechanism, in which each connection adjusts its sending rate based on implicit feedback from the network (the dropping of packets), has two potential problems. First, it relies on a property known as ack-clocking; the arrival of an acknowledgment triggers departures of new packets, which helps to smooth out the transmission of packets to a rate at which the network can accept them. In SSCOP, since ACKs are cumulative and only periodically sent, we must find another technique to smoothly send data. Second, it is unlikely that flow control within a satellite network will be completely distributed with no bandwidth management; at the very least, access bandwidth must be assigned at the MAC layer. The solution we propose is based on modifications to TCP's flow control. In particular, we describe a mechanism which adapts to the amount of rate control in the network, ranging from no rate control (a distributed system like TCP) or tight rate control (explicit rate control from the network).

We start with the basic TCP algorithm and describe operation when there is no rate control from the network. The protocol maintains a congestion window, which is set to an initial number of segments[4] and which is guaranteed never to exceed the offered window from the receiver. The protocol then undergoes slow start by increasing its congestion window by 1 packet for each packet which is acknowledged, until congestion is detected (either by loss or by explicit notification) or the receiver's offered window is reached. If congestion is detected, the congestion window is decreased by 1/2, and the protocol then moves into a phase of congestion avoidance, during which it is permitted to increase its window by C packets per RTT; C is a constant greater than or equal to 1. Upon subsequent detection of congestion, the congestion window is halved (no further reductions are made until the protocol fully recovers from the discovered loss), but slow start is never re-entered since there are no coarse timeouts. Note that the protocol has the opportunity to increase its window or enable new retransmissions only upon receipt of a STAT or USTAT message; therefore, at every reception of a STAT or USTAT, the transmitter counts how many transmissions are enabled, and schedules them to be sent uniformly over the estimated RTT of the connection. The estimated RTT is obtained by comparing the timestamp in a received STAT with the current time, and low pass filtering over many samples. We call this transmission timer the *delayed send* timer.

Next, consider the case in which a minimum and maximum sending rate are imposed by the network. The technique described above easily generalizes by constraining the allowable values of the timeout interval for the *delayed send* timer. If a hard upper bound exists on the sending rate, retransmissions can also be counted among the packets scheduled to be sent. The required granularity of the *delayed send* timer depends on the granularity of the rates enforced by the network and on the access speed of the uplink.[5]

[4] There has been a recent proposal to make the TCP initial window as large as 4 segments [Flo97].

[5] It may be possible to relax the required granularity of this timer if the MAC layer (probably implemented in hardware) also performs traffic smoothing.

STP protocol features	Origin	Modifications
Error recovery mechanism	SSCOP	Retransmissions delayed to account for possible misordering
Packet formats	SSCOP	Added checksum, port numbers, timestamps, notification of errors
Flow control	TCP	Delayed send timer added for traffic smoothing, rate control; no coarse timeouts
Connection control protocol logic	SSCOP	Enables sending of data upon connection start
Error detection	TCP/AAL 5, SCPS-TP	Receiver reports detected errors to transmitter; unlike TCP, checksum does not cover IP header

Table 1: Origins of STP protocol features.

Finally, the original protocol relied on the AAL type 5 CRC for error detection. If the underlying satellite link layer protocol does not incorporate a strong CRC, we recommend incorporating the AAL type 5 CRC into STP, since such a strong CRC is very important for error resilience over wireless networks which use FEC [Chi94]. Otherwise, a weaker checksum, such as the 16 bit IP checksum, can suffice. To enable better network layer mobility support, the STP checksum does not cover the IP header.

4.2 Additions for asymmetric and satellite networks

SSCOP is designed for bandwidth efficiency; therefore, if data transfer is only occurring in one direction, the reverse direction is only used for periodic status messages. Under normal operation, the protocol can in some cases operate over networks with a 1000:1 ratio between forward and reverse bandwidth. As a refinement for highly asymmetric channels, if the normal STAT/USTAT traffic is too much for the reverse channel, the receiver can periodically suppress STAT transmissions (at the potential cost of slowing down the forward data transmission). Another consequence of asymmetric networks is the high variability in RTT which is often encountered. Because the protocol logic does not require an accurate value of the RTT, it operates smoothly when such variability is present.

Another problem in the satellite environment is that the three-way handshake takes too long to occur before data transfer is enabled. It is possible to permit data transmission immediately after the first message. SSCOP incorporated the facility for data transmission during connection establishment, but the feature was never used. In STP, we make use of this capability. A similar capability exists in TCP for Transactions [Bra92].

One additional refinement for wireless networks which exhibit a high bit error ratio (BER) is distinguishing between lost and corrupted packets. At the receiver, if a received packet is found to be corrupted, the protocol can make a guess as to what sequence number it most likely was (in most cases by simply looking at the sequence number field). The “E(rror)” bit in the STAT and USTAT list fields can then be flagged to indicate to the transmitter that the requested packet was probably received in error and not dropped due to congestion. Of course, if the IP header is corrupted somewhere in the network, the errored payload will not make it to the transport layer. However, if the packet is large, most bit errors will hit the payload.

Table 1 summarizes the relationship between STP and other protocols. The *delayed send* timer implementation is new; other protocol features of STP can be found in other protocols.

5. Simulation results

We have implemented the data transfer procedures of STP in the UCB/LBNL Network Simulator *ns*.[6] This simulator is already in wide use by the networking research community, and is particularly well supported for TCP simulations. Our implementation and simulation scripts are also available.[7]

We were interested in comparing the performance of STP and TCP by examining the performance of persistent connections (i.e., long file transfers) over simulated satellite network topologies. In the following, we describe the simulation, present the results of a comparison of STP and TCP performance, and present

[6] http://mash.cs.berkeley.edu/ns/

[7] http://www.cs.berkeley.edu/~tomh/software.html

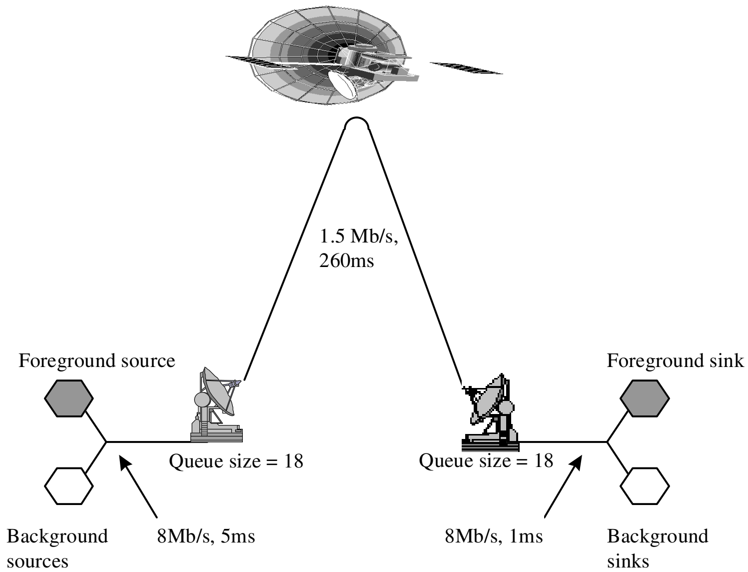

Figure 3a: Geosynchronous satellite simulation configuration.

performance results for STP connections in a high BER environment.

5.1 Simulation configuration

5.1.1 Topologies

Figure 3 illustrates the two simulated topologies with which we experimented. The first topology (Figure 3a) is configured to emulate a T1 channel over a geosynchronous (GEO) satellite channel with a one way delay of 260ms. The total RTT experienced by a user is 532ms, excluding queuing and transmission delays. The queue sizes were set to at least 10% of the outgoing line rate. Figure 3b illustrates a hypothetical low-earth-orbit (LEO) configuration. The access links to the satellite have a one-way propagation delay of 5ms and a bit rate of 2Mb/s, and the intersatellite links have a propagation delay of 10ms and a bit rate of 100Mb/s. The topology is similar to the proposed Teledesic system.

5.1.2 TCP configuration

Two TCP variants were tested; TCP with selective acknowledgments (TCP-Sack) and TCP "Reno" (corresponding to a current reference TCP implementation). For the TCP simulation configuration, we used *ns* defaults for all parameters except for the window size, which was opened up to a large value to avoid artificially constraining the sender, and the coarse timeout interval of 500ms, which is set by default in *ns* to the non-standard value of 100ms. Since delayed acknowledgments are standard practice in current implementations (to reduce the reverse channel bandwidth), we configured the TCP sinks to send delayed acknowledgments (typically, an ACK is sent for every 2 segments). For STP, we set the threshold on duplicate acknowledgments before a USTAT is sent)to 3 (as in TCP), and the number of POLLs per RTT was also set to 3, since in SSCOP it has always been recommended that multiple POLLs be sent each RTT if the RTT is long [Hen95].

5.1.3 Flow control

In order to permit a fair comparison of the basic protocol performance, we implemented a flow control policy in STP identical to that of TCP; namely, additive window increases of 1 packet per RTT and a multiplicative decrease by 1/2 during the congestion avoidance phase.[8] For both the GEO and the LEO topologies, we added background HTTP traffic which occupied, on average, about 10% of the bottleneck links. This traffic emulates actual WWW usage based on empirical distributions of actual traffic traces. The purpose was not to heavily congest the network (the behavior of the congestion avoidance mechanisms of the foreground connections guaranteed that congestion would frequently occur) but to add variability to the simulation runs in order to break

[8] This does not imply that we suggest this is an optimal mechanism for the satellite network. We discuss some fairness problems with this mechanism in [Hen97].

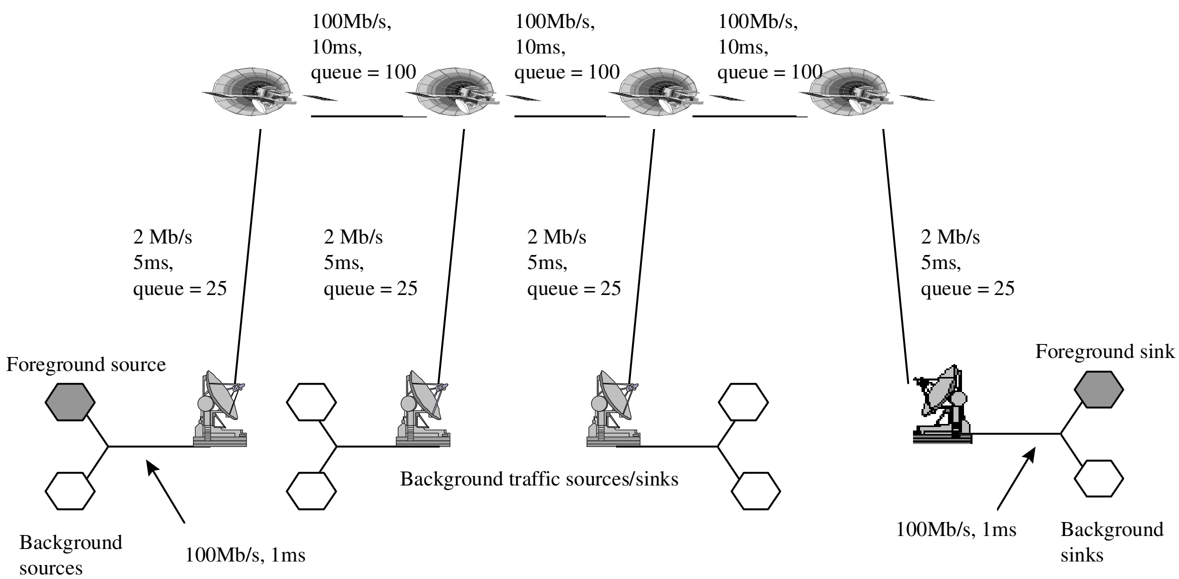

Figure 3b: Low earth orbit satellite simulation configuration.

up any phase effects. In the GEO topology, the background traffic was sent in the same direction as the foreground transfer, creating an occasional bottleneck at the queue at the ingress of the satellite network. In the LEO topology, the background traffic was sourced from multiple ground stations in different spot beams, creating an occasional bottleneck at either the ingress or the egress of the satellite network. In some situations, discussed below, we balanced the traffic load in each direction by creating a persistent TCP-Sack connection in the reverse direction along with the same amount of simulated WWW traffic, in order to cause periodic congestion losses in the reverse path as well.

5.2 Comparison of STP with TCP

Tables 2 and 3 present the results of an average of 200 simulation runs, each 60 seconds long, over the GEO and LEO topologies, respectively. We compared the performance of STP, TCP-Sack, and TCP-Reno, first with foreground traffic only and then with bidirectional traffic. Since no bit errors were introduced on the links, all losses were due to congestive losses induced by the congestion avoidance mechanisms. We used a fixed packet size of 1000 bytes (including TCP/IP or STP/IP overhead). The performance is compared in terms of forward throughput achieved ("goodput"), forward bandwidth efficiency (ratio of goodput to total TCP/IP data transferred in the forward direction) and reverse channel bandwidth usage. We observe the following:

- TCP-Sack and STP both significantly outperform TCP-Reno in the forward direction, especially over a long delay path. This is because both TCP-Sack and STP are able to recover multiple losses within a RTT.
- TCP-Sack uses a significant amount of bandwidth in the reverse channel for returning ACKs. TCP-Reno also uses much more reverse bandwidth than STP. For all of the simulations averaged together, STP required roughly 5 kb/s, while TCP-Reno needed 17 kb/s and TCP-Sack used 21 kb/s.
- For one way traffic, STP outperforms TCP-Sack in the GEO case, in terms of throughput in the forward direction. This is largely because of TCP-Sack's delayed acknowledgments, which cause the congestion window to open up more slowly than if every segment were acknowledged. If TCP-Sack were to acknowledge every segment, it would slightly outperform STP since, on average, acknowledgments are delayed 1/6 of the RTT due to the cumulative acknowledgment delay in STP. Doing so, however, would double the usage of the reverse channel bandwidth, which is already high with delayed acknowledgments. Another benefit to STP in a high BW*RTD environment with high losses is that it reports the complete state of the receiver with every STAT. In the LEO case, TCP-Sack slightly outperforms

	Throughput (kb/s)	95% conf. int. (+/-)	Fwd. efficiency	95% conf. int. (+/-)	Reverse bw (kb/s)	95% conf. int. (+/-)
One way traffic, GEO (Figure 3a)						
STP	672.3	20.8	0.938	0.001	2.2	0.03
TCP-Sack	594.9	20.5	0.945	0.001	13.2	0.3
TCP-Reno	296.5	15.2	0.928	0.003	6.7	0.3
Two way traffic, GEO (Figure 3a)						
STP	595.7	20.2	0.933	0.008	2.0	0.03
TCP-Sack	388.4	13.9	0.939	0.001	8.9	0.3
TCP-Reno	259.2	12.0	0.922	0.003	6.0	0.2

Table 2: Performance comparison of STP, TCP-Sack, and TCP-Reno over the GEO topology.

	Throughput (kb/s)	95% conf. int. (+/-)	Fwd. efficiency	95% conf. int. (+/-)	Reverse bw (kb/s)	95% conf. int. (+/-)
One way traffic, LEO (Figure 3b)						
STP	1668.2	13.8	0.961	0.001	8.5	0.05
TCP-Sack	1715.1	14.5	0.957	0.001	37.4	0.3
TCP-Reno	1552.9	11.6	0.957	0.002	33.5	0.2
Two way traffic, LEO (Figure 3b)						
STP	1440.5	13.8	0.960	0.001	8.0	0.04
TCP-Sack	1154.2	12.6	0.958	0.001	25.5	0.3
TCP-Reno	975.4	15.5	0.957	0.001	21.3	0.3

Table 3: Performance comparison of STP, TCP-Sack, and TCP-Reno over the LEO topology.

STP. We believe that this is due to STP slowing down its sending rate in response to a queueing backlog (which increases its RTT).

- STP outperforms TCP-Sack to a greater extent when there is two way traffic in the system. We believe that this is due to the effect of "ACK compression," which disrupts TCP's self clocking behavior. STP is relatively insensitive to these effects, and in fact it seems that much of the reduction in its performance is due to the presence of the reverse (TCP-Sack) connection's acknowledgments in its forward path buffers, effectively decreasing its usable buffer space.
- TCP-Sack and STP have comparable performance in terms of forward bandwidth efficiency. Although STP has less per-packet overhead, the overhead of the 3 POLL messages per RTT must be amortized across data; therefore, the efficiency improves as the throughput improves.
- The difference in performance between the protocols was reduced in the LEO case as compared to the GEO case. There is less of an advantage in using TCP-Sack instead of TCP-Reno when the RTT is smaller.

In summary, when using the standard TCP flow control policies in identical simulated satellite environments, STP generally outperformed TCP-Sack and TCP-Reno in terms of throughput while using much less bandwidth in the reverse channel. We did not explore possible improvements via fine-tuning of the protocols.

5.2 STP performance in a high BER environment

We next examined the performance of STP in an environment in which the network controls the maximum rate at which the protocol can send, but in which the BER is varied from $1e^{-4}$ to $1e^{-7}$. The modification to STP's flow control is simple. The *delayed send* timer is regularly scheduled to expire at the rate at which packets are allowed into the network. If retransmissions are queued, they are sent with highest priority in the scheduled slot; otherwise, a new data packet is sent. There is no rate control imposed on the reverse direction.

Figure 4 illustrates results for a 1 Mb/s connection (1 Mb/s allowable transport bandwidth), again using the GEO topology shown in Figure 3a, but for which bit errors are randomly inserted on the link.[9] Again, the IP packet sizes are 1000 bytes for data traffic. Since the STP/IP

[9] Actual wireless links tend to exhibit burst errors, which would shift the curves leftward by roughly the burstiness factor.

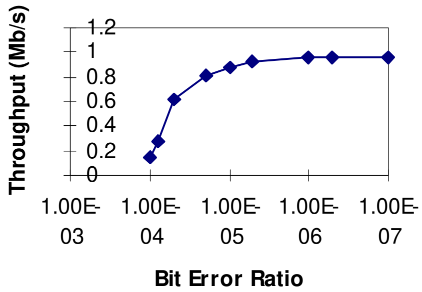

Figure 4a: Forward throughput performance of STP on a 1 Mb/s channel.

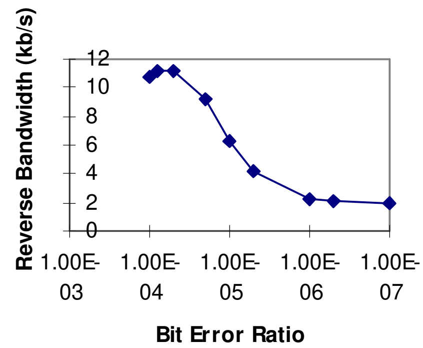

Figure 4b: Reverse channel bandwidth required for STP as a function of BER.

overhead per packet is 32 bytes, the usable throughput is constrained to be 968 kb/s at best. Figure 4 illustrates that the selective retransmission mechanism provides high efficiency even as the BER degrades substantially, and that the reverse channel bandwidth also rises as the BER increases. Better performance could be obtained as the BER degrades by using smaller packets (since with a BER of $1e^{-4}$, the packet loss rate is 55% with 1000 byte packets). We did not compare STP with TCP in this case since TCP has no facility for explicit rate control.

6. Conclusions

We have reported on the adaptation of an ATM based protocol known as SSCOP for the general satellite network environment, including networks based on connectionless network service. We refer to this modified protocol as the "Satellite Transport Protocol" (STP). In particular, we have incorporated a flow control mechanism useful for either rate or window control, updated the protocol logic to allow for misordered data in the network, and added some features for operation in a wireless environment. The protocol is fundamentally based on periodic full state exchanges between the transmitter and the receiver. It operates very efficiently in a satellite environment even when the connection BER or the RTT variability is high. It also is naturally designed to operate well over networks exhibiting bandwidth or delay asymmetries in the forward and reverse channels. We have demonstrated the basic performance of this protocol using simulation.

We envision that STP or similar variant could be used either internally within a satellite network for control traffic or as the satellite portion of a split TCP connection. For example, control traffic between a satellite terminal and a network control center will need to be reliably delivered; STP is a good candidate because it is very bandwidth efficient and relatively insensitive to long RTTs or asymmetric network topologies. We would also like to encourage the use of similar protocol design principles for future applications which implement their own error recovery techniques, following the principles of Application Level Framing [Cla90], so that such applications work well without modification in a satellite environment.

We are working on a UNIX kernel implementation of STP and intend to perform experiments over actual satellite links. Of particular interest is the overhead associated with running a fine grained *delayed send* timer in the kernel. We also believe that the protocol could be made more self adaptive to its environment, such as by adapting the POLL frequency, the duplicate acknowledgment threshold, the packet size, and rate thresholds based on observed performance. It should also be a simple modification to allow partial loss recovery, for applications which may want reordering and a nominal wait for late packets, but which do not want to wait an excessive amount of time for error recovery.

Acknowledgments

This work was supported by DARPA contract DAAB07-95-C-D154, by the State of California under the MICRO program, and by Hughes Aircraft Corporation, Metricom, Fuji Xerox, Daimler-Benz, and IBM.

References

[All96] M. Allman, H. Kruse, and S. Ostermann, (1997). "An Application-Level Solution to TCP's Satellite Inefficiencies," *Proceedings of the First International Workshop on Satellite-based Information Services (WOSBIS)*, Rye, NY, November.

[ANS94] American National Standard T1.637, (1994). "B-ISDN Signaling ATM Adaptation Layer--Service Specific Connection Oriented Protocol (SSCOP)."

[Bak95] A. Bakre and B. Badrinath, (1995). "I-TCP: Indirect TCP for Mobile Hosts," *15th Int'l Conference on Distributed Computing Systems (ICDCS)*, May.

[Bal95] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, (1995). "Improving TCP Performance over Wireless Networks." *1st ACM International Conference on Mobile Computing and Networking (Mobicom)*, November.

[Bal97a] H. Balakrishnan, V. Padmanabhan, and R. Katz, (1997). "The Effects of Asymmetry on TCP Performance over Wide-Area Wireless Networks," *to appear, 3rd ACM International Conference on Mobile Computing and Networking (Mobicom)*, September.

[Bal97b] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz, (1997). "TCP Behavior of a Busy Web Server—Analysis and Improvements," submitted to *IEEE Infocom '98*.

[Bla93] U. Black, (1993). *Data Link Protocols*, Prentice Hall, Englewood Cliffs, NJ.

[Bra92] R. Braden, (1992). "Extending TCP for Transactions-- Concepts," *Internet RFC 1379*.

[Cla90] D. Clark and D. Tennenhouse, (1990). "Architectural Consideration for a New Generation of Protocols," in *Proceedings of ACM SIGCOMM '90*, Philadelphia, PA, September.

[Chi94] D. Chitre, D. Gokhale, T. Henderson, J. Lunsford, and N. Mathews, (1994). "Asynchronous Transfer Mode (ATM) Operation via Satellite: Issues, Challenges, and Resolutions," *International Journal of Satellite Communications*, Vol. 12, pp. 211-222.

[Dur96] R. Durst, G. Miller, and E. Travis, (1996). "TCP Extensions for Space Communications," *2nd ACM International Conference on Mobile Computing and Networking (Mobicom)*, November, pp. 15-26.

[Flo97] S. Floyd, (1997). Message to end2end-interest group mailing list, February 7.

[Hen95] T. Henderson, (1995). "Design Principles and Performance Analysis of SSCOP: A New ATM Adaptation Layer Protocol." *ACM Sigcomm: Computer Communications Review*, April.

[Hen97] T. Henderson, E. Sahouria, S. McCanne, and R. Katz, (1997). "Improving Fairness of TCP Congestion Avoidance," submitted to *IEEE Infocom '98*.

[ITU94] ITU-T Recommendation Q.2110, (1994). "B-ISDN Signaling ATM Adaptation Layer-- Service Specific Connection Oriented Protocol (SSCOP)," Geneva.

[JBB92] V. Jacobson, R. Braden, and D. Borman, (1992). "TCP Extensions for High Performance," *Internet RFC 1323*.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, (1996). "TCP Selective Acknowledgment Options," *Internet RFC 2018*.

[Net90] A. Netravali, W. Roome, and K. Sabnani (1990). "Design and Implementation of a High-Speed Transport Protocol." *IEEE Transactions on Communications*, Vol. 38, No. 11, pp. 2010-2024, November.

[Pax97] V. Paxson, (1997). "Automated Packet Trace Analysis of TCP Implementations," to appear, *Proceedings of ACM SIGCOMM '97*, Cannes, France, September.

[Pos81] J. Postel, (1981). Transmission Control Protocol. *Internet RFC 793*.

[Zha97] Y. Zhang, D. DeLucia, B. Ryu, and S. Dao (1997). "Satellite Communications in the Global Internet: Issues, Pitfalls, and Potential," *INET '97*, Kuala Lumpur, Malaysia, June.