SCPS-TP, TCP, AND RATE-BASED PROTOCOL EVALUATION

Diepchi T. Tran, Frances J. Lawas-Grodek,
Robert P. Dimond, and William D. Ivancic
NASA Glenn Research Center

SCPS-TP, TCP, AND RATE-BASED PROTOCOL EVALUATION

CONTENTS

SCPS-TP, TCP, AND RATE-BASED PROTOCOL EVALUATION

1 ABSTRACT

Tests were performed at Glenn Research Center (GRC) to validate the operation of Space Communications Protocol Suite-Transport Protocol (SCPS-TP) relative to the Consultative Committee on Space Data Systems (CCSDS) specification, to perform a comprehensive comparison of SCPS-TP protocol options to IP based protocols, and to determine the implementation maturity level of these protocols – particularly for higher speeds. The testing was performed over reasonably high data rates of up to 100 Mbps with delays that are indicative of near-planetary environments. The tests were run for a fixed packet size, but for various errored environments. The results indicated that SCPS-TP congestion-friendly options perform slightly better than TCP SACK protocols at moderate and high error-rates whereas IP rate-based protocols performed slightly better than SCPS-TP rate-based options. The results also show that existing standard transport protocols and capabilities (drawn from a variety of communities) appear to satisfy all known near-planetary mission needs.

This report documents the testing performed to validate operation of SCPS-TP relative to the specification and provides a comprehensive comparison of SCPS-TP protocol options relative to IP based protocols.

2 EXECUTIVE SUMMARY

There have been numerous debates regarding the actual improvements that SCPS may provide over the ever-evolving TCP/IP protocol suite. In addition, much of the SCPS initial testing and demonstrations did not provide what many consider to be a valid comparison relative to TCP as known improvements to TCP for long bandwidth-delay networks were often not implemented (i.e. large windows, selective acknowledgements) [1]. Other testing was performed over simulated links where SCPS would provide little advantage due to the very low bandwidths [2]. Some well documented and thorough testing has been performed at lower rates. These results correlate well with our test results [3,4].

In order to gain a better understanding of the actual improvements, if any, that SCPS could provide relative to TCP and to determine the maturity of the various protocols for higher-rate links, the NASA Space and Data Communications Systems (SCDS) Office requested Glenn Research Center to perform a comprehensive set of tests. Tests were performed at Glenn Research Center to validate the operation of SCPS-TP relative to the CCSDS specification, and to perform a comprehensive comparison of SCPS-TP protocol options relative to IP based protocols. Note, this testing was only performed for

SCPS Transport Protocol (SCPS-TP). Neither the security mechanisms nor networking protocol was implemented or tested[1].

We have studied the effect of delay and BER on the performance of congestion-friendly and rate-based protocols in uncongested and limited congested emulated space links. The results correlate well with other testing of SCPS-TP and TCP.

- The single stream and multi-stream test results clearly illustrate that the SCPS-Vegas enhancements to TCP provide measurable performance improvements over the TCP SACK implementation tested in space-based environments. The value of these performance increases is subjective and would need to be judged on a mission-by-mission basis.
- Very small transactions such as command and control should see little difference in performance for TCP or any variant of SCPS-TP or a rate-based protocol.
- In extremely error-prone environments with high RTT delays, a rate-based protocol is advisable assuming one properly engineers the network. However, one must beware of using rate-based protocols on shared networks, unless one is operating the protocol in an environment where bandwidth reservation is practical and available.
- The deployment of an in-kernel protocol may be more desirable than the deployment of an application level protocol, for more efficient use of resources and performance issues. However, one may also argue, that it is far easier to maintain a protocol at the application level than within a kernel.
- Even with equal performance, the SCPS rate-based protocol may more desirable to implement than other rate-based protocols such as the multicast Dissemination Protocol, as SCPS is capable of requiring only sending-side only modification.
- The existing standard transport protocols and capabilities (drawn from a variety of communities) appear to satisfy all known mission needs; however, the space community should maintain an awareness of current and future TCP research. New TCP research may dramatically improve TCP operation for near-*planetary* environments. Some pertinent areas include Stream Control Transmission Protocol (SCTP), TCP Pacing with Packet Pair Probing, TCP Westwood, and TCP Explicit Transport Error Notification (ETEN).

3 INTRODUCTION

In the late 1980's and throughout the 1990's, the Internet has rapidly developed allowing vast improvements in communication and networking. These technologies utilize packet-based communications rather than circuit-based communications. The Consultative Committee on Space Data Systems (CCSDS) foresaw the need to take advantage of this new Internet technology and developed the Space Communications Protocol Suite (SCPS) to address some specific issue related to space systems. Thus, the TCP/IP protocol suite was investigated and modifications to the networking, security and

[1] SCPS-Network Protocol was not evaluated in this study due to lack of hardware and software implementations. All routing was performed over IPv4, which is deployed throughout the Internet. Note that SCPS-NP will not accommodate the National Security Agency's High Assurance Internet Protocol Interoperability Specification (HAIPIS); thus, use of SCPS-NP for secure government applications may be problematic.

transport protocols were specified. These specifications are known as the SCPS Security Protocol, Network Protocol, Transport Protocol, and File Transfer Protocol (SCPS-SP, SCPS-NP, SCPS-TP, and SCPS-FP).

There have been numerous debates regarding the actual improvements that SCPS may provide over the ever-evolving TCP/IP protocol suite. In addition, much of the SCPS initially testing and demonstrations often did not provide what many consider to be a valid comparison relative to TCP as known improvements to TCP for long bandwidth-delay networks were often not implemented (i.e. large windows, selective acknowledgements) [1]. Other testing was performed over simulated links where SCPS would provide little advantage due to the very low bandwidths [2]. Some well documented and thorough testing has been performed at lower rates. These results correlate well with our test results [3,4].

In order to get a better understanding of the actual improvements, if any, that SCPS could provide relative to TCP and to determine the maturity of the various protocols for higher-rate links, the NASA Space and Data Communications Systems (SCDS) Office requested Glenn Research Center to perform a comprehensive set of tests. This report documents the comprehensive set of tests performed to validate the operation of SCPS-TP (relative to the CCSDS specification), and to provide a comprehensive comparison of SCPS-TP protocol options to other TCP based protocols. This testing was only performed for SCPS Transport Protocol (SCPS-TP). Neither the SCPS-SP security mechanisms nor the SCPS-NP networking protocol was implemented or tested.

4 PROTOCOL OVERVIEW

4.1 TCP

TCP is a reliable transport protocol, which uses a sliding-window based congestion control algorithm proposed by Van Jacobson and others. In particular, TCP congestion control methods include: slow-start [5], congestion avoidance, fast retransmit and fast recovery algorithms [6]. The slow-start algorithm is activated (triggered) at the beginning of a transfer or after a retransmission timer timeout (RTO). Slow-start occurs until the congestion window (*cwnd*) reaches the slow-start threshold (*ssthresh*) or if packet loss occurs. During the slow-start phase, if the receiver buffer size is large enough, the number of segments injected into the network is doubled every Round Trip Time (RTT). When the *cwnd* exceeds the *ssthresh*, the congestion avoidance algorithm is used to lower the sending rate by increasing the *cwnd* by at most one segment per RTT. This is the additive increase algorithm of TCP and is done to probe for additional network capacity. Upon the arrival of three duplicated acknowledgements (ACKs) at the sender, the fast retransmit algorithm is activated which retransmits that segment without waiting for the RTO to expire. Duplicate acknowledgements may occur when a packet is lost yet three additional packets arrive at the receiver. After the retransmission of the lost segment, the fast recovery method is used to adjust the *cwnd*. As a result *ssthresh* is set to half the value of *cwnd*, and then the *cwnd* is cut in half plus three segments. At this point, for each duplicate ACK that is received, the *cwnd* is increased by one segment until the ACK of the retransmission arrives. After that, *cwnd* is set to *sshthresh* and the

additive increase algorithm is activated until either is equal to the advertised receiver window or until loss is detected, indicating possible congestion.

Since the above fast retransmit method can only fix one lost segment per RTT, the subsequent lost segments within that RTT usually have to wait for the RTO to be expired before being resent. In addition, an aggressive sender can retransmit segments that may have been received. The combination of the SACK [7] option and fast retransmit/fast recovery algorithms can be used to solve these problems. With the SACK and timestamp options, the receiver informs the sender about segments that have been received such that the sender can determine up to three segments which have been lost within an RTT.

For most variants of TCP currently used in practice including TCP Reno and TCP SACK, the sending rate is cut in half each time a loss occurs. The sending rate is then gradually increased until another loss happens. This process is known as Additive Increase, Multiplicative Decrease (AIMD) is repeated until all of the data has been transmitted. This is one of the reasons TCP has difficulty operating *efficiently*[2] over long delay, error-prone networks.

4.2 SCPS

4.2.1 Van Jacobson (VJ)

SCPS-VJ has the same congestion control mechanism as described above for TCP except that SCPS-VJ uses the Selective Negative Acknowledgment (SNACK) [8] option which is adapted from NACK, Negative Acknowledgment [9]. SNACK identifies specific lost segments that need to be retransmitted, and can inform the sender of multiple lost packets at a later time in a bit-efficient manner. In addition, since SNACK does not depend on the fast retransmit algorithm to resend the lost segments, the sender does not need to wait for the three duplicated ACKs which may never arrive in high BER environment or in an asymmetric network with an extremely slow return link.

4.2.2 Vegas

Under this section, the original TCP-Vegas proposed by L. Brakmo and L. Peterson [10] is presented, and then the modifications to this code is described as incorporated under SCPS-Vegas.

Whereas the VJ method saturates the network and uses the loss segment as an indication of congestion, TCP-Vegas tries to avoid congestion in a network before it experiences losses. The Vegas algorithm tries to predict the congestion by monitoring the variations of the throughput and adjusts the *cwnd* based upon this throughput measurement. As a result, the sending rate in Vegas can be reduced before losses occur.

The following descriptions are of retransmission, congestion avoidance, and modified slow-start mechanisms as used in TCP-Vegas:

[2] TCP will reliably transfer all data over a long delay, error-prone network. However, for current TCP deployments, a single TCP flow will not fully utilize the available bandwidth under such conditions.

Retransmission: First, Vegas records the system clock each time a segment is sent and its corresponding ACK arrives. Then, Vegas uses these times to calculate the RTT of the segment and a timeout period for that segment based upon this RTT. Second, when a duplicated ACK is received, Vegas retransmits the segment if its timeout period has expired without waiting for three duplicated ACKs to arrive. Third, when a non-duplicated ACK of the first or second segment after the retransmission arrives, Vegas checks the timer of this segment again to see if it is expired. If it has, Vegas retransmits the segment. This retransmit helps to "fix" any other segments which may have been lost before the retransmission without waiting for duplicated ACK. In addition, in the case of multiple losses, the *cwnd* in Vegas is reduced only when the dropped segment was sent after last decreasing of *cwnd*. The losses that occur before the *cwnd* reduction do not imply that the network congestion happen for the current *cwnd* size. As a result, no further decrease in *cwnd* size is needed.

Congestion Avoidance: TCP-Vegas adjusts the *cwnd* based on the difference between the expected throughput and the actual throughput (difference = expected – actual). Also, two thresholds, *alpha* and *beta* (*alpha* < *beta*), are defined as indicators of too little and too much extra data in the network respectively. If the difference is less than *alpha*, which indicates that the network capacity is large enough to achieve the expected throughput, the *cwnd* is increased linearly during the next RTT. If the difference is greater than *beta*, which implies a sign of congestion occurring in the network, the *cwnd* should be decreased linearly during the next RTT. If the difference is between *alpha* and *beta*, the *cwnd* remains unchanged.

Modified Slow-Start: Vegas allows the *cwnd* to double its value only every other RTT in order to detect and avoid congestion during the slow-start phase. Vegas leaves the slow-start and enters linear mode when the actual throughput falls below the expected throughput by the equivalent of one router buffer or when loss has occurred.

In the SCPS-RI version 1.1.62 implementation, MITRE made several modifications to the above original slow-start and congestion avoidance mechanisms of L. Brakmo and L. Peterson:

- [10] indicated that the Vegas-Congestion control window (*cwnd*) can increase exponentially only every other round trip time (RTT) during slow-start phase. The congestion control window in the slow-start phase of SCPS-Vegas was changed in the 1.1.62 implementation to double upon the arrival of the first acknowledgement in every RTT, ensuring that the *cwnd* would grow exponentially every RTT (equivalent to the slow-start process in SCPS-VJ and TCP).

- In the congestion avoidance phase, the *cwnd* in the original Vegas algorithm is decreased by one packet when the difference between the expected throughput and actual throughput is greater than *beta*. In SCPS-Vegas, the *cwnd* is reduced by half of the amount of packets when the difference is greater than *beta*.

SNACK and delayed ACKs were enabled while testing this SCPS-Vegas option.

There is also a SCPS-Vegas-Corruption option, which has the same retransmission, congestion avoidance, and slow-start algorithms as in the SCPS-Vegas-Congestion option except for the protocol's reaction to packet loss. The *cwnd* under SCPS-Vegas-Congestion is reduced by half in response to a packet dropped, while the *cwnd* in SCPS-Vegas-Corruption remains unchanged.

4.2.3 Pure Rate Control

The SCPS Pure Rate Control option does not activate the congestion control algorithm. The sending rate depends on the value of the rate option defined by the user and the receiver buffer size. Like the TCP and SCPS-VJ tests, the acknowledgments for this test are delayed ACKs. The SNACK option, as described previously, was also enabled in these Pure Rate Control tests.

There is an additional option of the SCPS Pure Rate Control using "Strictly Delayed ACKs". Here, the ACKs are sent back every time interval as defined in the Delayed ACK timer instead of sending an ACK every packet or every other packet. Using the Delayed ACK timer as a trigger to send back an ACK can be beneficial in a long delay environments, where a longer period of time can pass before a second packet can arrive.

4.3 MDPv2

The original Multicast Dissemination Protocol (MDP) [11] was developed between 1995-1997 as the underlying protocol for the Image Multicaster (IMM) [12], a reliable multicast application developed in 1993, also used on the Multicast Backbone (MBONE) for delivery of compressed image files and some bulk data content to multicast receivers. The protocol is UDP-based, providing high throughput by avoiding congestion control mechanisms in favor of a user selectable rate. MDP achieved its reliability through the use of NACKs, and can scale to provide efficiency in large multicast groups through NACK suppression (to minimize receiver message implosion) and the aggregation of control messages. In addition, MDP's method of operation adapts well to asymmetric links.

In 1997 Multicast Dissemination Protocol Version 2 (MDPv2) [13] was developed, extending MDP capabilities to include a parity-based repair mechanism, an emission control (EMCON) mode where clients refrain from message transmission, congestion control algorithm, and tunable parameters, allowing MDP to adapt to a myriad of networking environments.

GRC's testing utilized MDPv2 (1.9a4), which was compiled from source with increased receive buffers. The application was used in pure-rate mode without congestion control or forward error corrections in a unicast network environment. Rate for the application was set at its optimum, which was determined to be 40 Mbps.

4.4 MFTP

First submitted to the IETF in 1998 by Starburst Communications Inc, the Multicast File Transmission Protocol (MFTP) [14] consists of two protocol components: the Multicast Control Protocol, and the Multicast Delivery Protocol. Multicast Control Protocol is an administrative protocol allowing the server to dynamically control the joining and leaving of its multicast groups. The Multicast Delivery Protocol handles the reliable transmission of data to the registered clients.

Much like its MDPv2 counterpart, MFTP utilizes a rate-controlled UDP/IP stream to transport data combined with a non-acknowledgement scheme in order to provide TCP's reliability while bypassing TCP's congestion control cost. Unlike MDPv2, MFTP does not provide any parity-based repair scheme or optional congestion control.

NASA-GRC initially tested the MFTP application in support of STS-99 to provide bulk data transfers for the German Aerospace Agency, DLR. GRC's testing utilized MFTP version 3.05 with the rate set at the applications maximum of 15 Mbps. [15]

5 Test Bed Configuration, Procedures, and Options

5.1 General Configuration For All Testing

The GRC test bed environment (Figure 1) consists of two separate networks, representing a terrestrial and space network. The two networks are interconnected via several unique ATM virtual circuits (VCs), passing through an Adtech SX/14 channel simulator. The SX/14 allows for the insertion of time delays and random bit errors into the network flow. The networks on each side of this channel consist of a CISCO 7100 router (connected to the SX/14 via ATM), and a CISCO 2900 Catalyst ethernet switch (connected to the routers via fast ethernet). The Catalyst switches serve as our LANs, connecting to transfer originators, receivers, or analyzers for our tests.

Hosts on these switches are connected either to an active port, to allow the system to participate in a traffic flow, or to a mirrored port to allow the system to analyze traffic to and from a specific host. In addition, hosts on each LAN are configured with two physical interfaces. The first interface is required for external access with the host on our test bed in order to avoid impeding on traffic flows or analysis. The second interface on each host serves the test bed, and has multiple logical IP network interfaces with explicit route statements, to "force" IP traffic to take a specific route through one of the aforementioned unique VCs interconnecting our test bed networks.

All tests utilized well-known software programs such as "tcpdump", "tcptrace", and "xplot", which were installed on the NetBSD monitoring/tracing machines as the main tools to capture, analyze, and observe the performance of our test protocols. Tcpdump can be obtained from ftp://ftp.ee.lbl.gov/tcpdump.tar.Z, and tcptrace and xplot are both

available at http://jarok.cs.ohiou.edu/software . MITRE also modified the tcpdump and tcptrace, which we called "scps_tcpdump" and "scps_tcptrace", that served as the tools for monitoring the SCPS-TP protocol tests. It should be noted that a modified version of libpcap was needed from MITRE to compile with scps_tcpdump in order to use some options of tcpdump and debugging in SCPS.

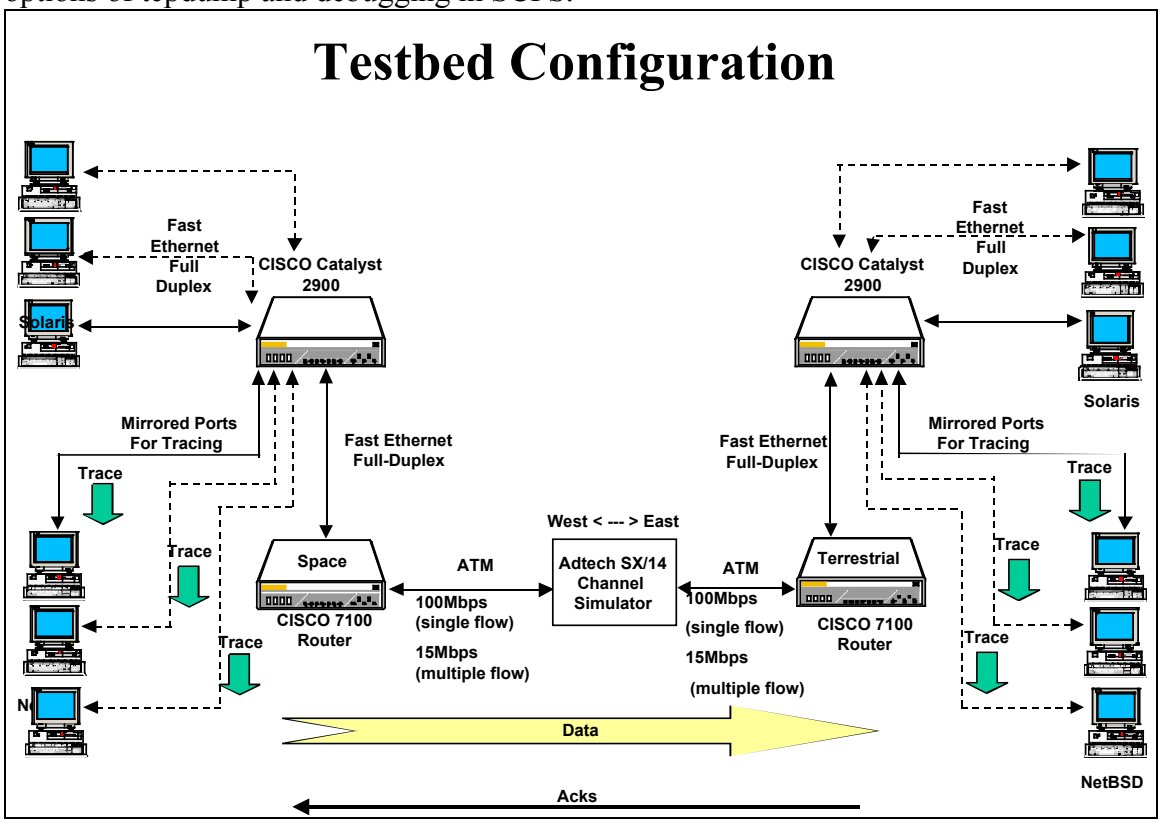


Figure 1. Test Bed Facility

5.2 Single-Flow Configurations

The single flow tests were for providing a baseline comparison of congestion-friendly TCP versus SCPS-TP performance, and for the evaluation of rate-based protocols devoid of congestion control mechanisms. During single-flow operation, two Sun machines (one on each network) running Solaris 7 were configured in full duplex mode, acting as either the receiver or the originator of a transfer session. Transfers were between the "terrestrial-single-rate" and "space-single-rate" logical networks, forcing the single traffic flow to traverse our space channel simulator through an ATM VC rate-limited to 100 Mbps bandwidth. To monitor and capture the network data, a PC running NetBSD 1.5 was connected to a mirrored port for each Sun system.

Software for the single-flow tests consisted of the following: the TCP protocol that came with the Solaris 7 kernel, which was used for the TCP-SACK test. The SCPS Reference Implementation (SCPS-RI versions 1.1.51, 1.1.62, and 1.1.66), provided by the MITRE Corporation, was used for the SCPS-TP tests. MDP testing utilized a GRC-specific compilation of the MDP 1.9a4 source code. All detailed settings of TCP, SCPS-TP, and MDP parameters for each test will be given in the Testing Results Section and

Appendices A through F. For baseline TCP testing, the popular "Test TCP Program" (ttcp) was used, which was originally developed by the US Army Ballistics Research Lab (BRL), to determine TCP performance. A version of ttcp modified by MITRE, called "scps_ttcp", was used as the benchmarking tool for the SCPS-TP performance. The scps_ttcp source is included in the SCPS-RI software package.

Single-flow testing procedures required a minimum of multiple 1024 byte packet transfers for each series of tests in order to determine average protocol measurements and their deviations. A series consisted of manipulating Bit Error Rate (BER) (Possible: 0e0, 1e8, 1e7, 1e6, 1e5, and 1e4), round-trip transmission delay (Possible: 0 ms, 10 ms, 250 ms, and 500 ms), and file sizes (Possible: 100 KB, 1 MB, 10 MB, 100 MB), creating up to 96 different series of tests to conduct. For this reason, scripts were created to automate the following tasks:

1. Remotely set the BER and delay on the SX/14 channel simulator for each series of tests.
2. Login to data collection systems via ssh, initiating the appropriate data collection process.
3. Login to the sender and receiver of a protocol transfer, initiating the file transfer
4. Monitor for end of transfer, kill collection processes, and save data collected.
5. Repeat steps 2-4 multiple times for each series of tests.

Additional series consisted of the many SCPS options exercised such as Van-Jacobson Congestion, Vegas Congestion, Vegas Corruption, Pure Rate Delayed ACKs, and Pure Rate ACK Every Other Packet, for a total of up to 480 series of SCPS-related tests. Each series of tests consisted of 10-30 file transfers. The sheer number of tests to be performed for all evaluated protocols ran into over 4000 wall-clock hours of test bed utilization.

Because GRC personnel were not working with thoroughly established and tested protocols, the results could not be taken at face value. This necessitated the need for preliminary analysis of the collected data for each series of tests, to insure the protocols were performing within the realm of expected behavior.

5.3 Multiple-Flow Configurations

The network architecture and procedures for multi-flow testing is similar to that of the single-flow testing, with the exception that there are now three sender/receiver machines on each side of our aforementioned terrestrial-space internet. Each of these machines either originates or receives a transfer session between the "terrestrial-multi-rate" and "space-multi-rate" logical networks, forcing all traffic to traverse our space channel simulator through an ATM VC, rate limited to 15 Mbps bandwidth. This lower rate VC was used to insure that congestion would occur. In addition to the active participant changes, three monitor/capture machines were also required on each of our networks, as a machine using a mirrored port can only monitor traffic to and from one host in a switched environment. The dashed lines in Figure 1 connect those additional machines needed for the multiple flow tests.

In addition to the software used in previous testing, for the TCP-SACK tests, the TCP included with the Solaris 7 kernel was used in the first two pairs of sender/receiver

machines, and the TCP of the Solaris 8 kernel was used in the third pair. SCPS_RI version 1.1.62 was used in the SCPS-VJ and SCPS-Vegas-Congestion multiple flow tests.

As in single flow testing, file transfers and data collection were performed, with the average throughput and standard deviation calculated from the test results. Since there are three pairs of sending and receiving machines, there are six possible combinations of sending orders among the three senders. These combinations were picked randomly using the output of a random number function. In addition, by using the same random function, each sender was also randomly started from one to eight seconds apart from each other.

More information on each component of the single flow and multiple flow test bed are given in Appendix I.

5.4 Defined Protocol Options

5.4.1 Options Used For All Protocols:

The following are a summary of the *possible* variables and options used in TCP, SCPS-TP, MDP, and MFTP protocol single flow tests:

1. File sizes: 100 KB, 1 MB, 10 MB, 100 MB
2. Packet size: 1024 Bytes (except for MFTP which was tested with a 1472 byte packet size)
3. Delays: 0 ms, 10 ms, 250 ms, 500 ms
4. BERs: zero (baseline), 1e-8, 1e-7, 1e-6, 1e-5, 1e-4

5.4.2 TCP Option: SACK

5.4.3 SCPS Options for SCPS tests:

1. Van Jacobson Congestion Control, ACKing Every Other Packet (SCPS-VJ).
2. Pure Rate Control , ACKing Every Other Packet (SCPS-Pure Rate Control, Option F2).
3. Pure Rate Control, Strictly Delayed ACKs (SCPS-Pure Rate Control, Option F0).
4. Vegas Congestion Control, ACK 1/2 Packets, Assume Congestion (SCPS-Vegas-Congestion)
5. Vegas Congestion Control, ACK 1/2 Packets, Assume Corruption (SCPS-Vegas-Corruption)

5.4.4 Multiple-Flow Options

The multiple flow testing for TCP-SACK, SCPS-VJ, and SCPS-Vegas-Congestion was performed using the same packet size as listed in 5.4.1 plus the following options:

1. File size fixed at 50 MB
2. Round trip delay fixed at 500 ms
3. BER values limited to zero, 1e-7, and 1e-5
4. Random interval between start of competing flows

5.4.5 MDP Options

1. No Parity/Forward Error Correction (server)
2. Initial Transfer of file (and repairs) only (server)
3. Rate set to 40 Mbps (server)
4. No post processing of data (client)
5. Archive data (client)

5.4.6 MFTP Options

1. Maximum Datagram Unit set to default of 1472 (server)
2. Bound to TX & RX to unicast interface (client & server)
3. Initial file transfer (and repairs) only (server)
4. Transfer timeout set to 10000% (100 times) of first pass time.

6 THEORETICAL THROUGHPUT

In order to gain a feel for the overall performance of the various protocols, we calculated some theoretical bounds for TCP and pure rate-based protocols. In general, these calculations are for very large transfers that have reached a steady state conditions. These theoretical bounds are not indicative of small file transfers or small transactions that utilize a few packets such as command and control transactions. Appendices G and H list the theoretical throughput for both rate-based and congestion-friendly protocols.

6.1 Congestion Control-Based Protocol

The maximum theoretical throughput for congestion control-based protocols (TCP-SACK, SCPS-VJ and SCPS-Vegas-Congestion) running over an errored link is calculated using formula (1) from Mathis [16]. Note, this formula assumes the system has reached steady state (i.e. extremely large file transfers, not command and control transactions)

$$\text{Bandwidth} = 0.93*\text{MSS}/\text{RTT}*\text{sqrt}(p) \quad (1)$$

Where
MSS = maximum segment size
RTT = Round Trip Time
p = packet error rate

In our experiments, the user data packet size was set at 1024 Bytes[3], which was then used as MSS in formula (1).

For an error-free environment, the maximum throughput is equal to the receiver window divided by the RTT as given in formula (2). Note, this formula assumes sufficiently large transfers such that time that the transfer spends in slow-start is insignificant.

$$\text{Maximum Throughput} = (\text{Window size}/\text{RTT}) \quad (2)$$

[3] The 1024 was determined to be close enough to model ethernet packets, and is the default used in Berkeley's Network Simulator (NS)

Window sizes used for this TCP-SACK testing were 250 KB, 2.85 MB, and 5.7 MB for 10, 250, and 500 ms delays, and were applied in equation (2).

In order to count the effect of overhead on the throughput, it is assumed that the overhead is 58 Bytes (20 Bytes of TCP header, 20 Bytes of IP header, and 18 Bytes of Ethernet header). Thus, the maximum throughput is decreased by a factor of 1024/(1024+58).

6.2 Rate-Based Protocols

As a first order approximation for a rate-based protocol, the total transfer time of a file will equal the time needed to transmit the original packets of that file plus the time required to resend the dropped packets (if there are any dropped packets) and one round-trip-time which is used for the three-way-hand-shake at the beginning of a connection. The maximum throughput can be calculated by dividing the file size by the total transfer time of that file. It is assumed that every dropped packet can be fixed in the first retransmission. To include the effect of the overhead on the throughput, the maximum throughput is decreased by (1024/(1024+58)) as used in the throughput calculation for congestion-friendly protocols. The following is the formula that was used to compute the throughput of pure-rate control protocol for error links:

$$\text{Throughput(Mbps)} = \frac{(1024)*\text{File size} * 8}{(1024+58)*((\text{File size}*8*p/R) + (\text{File size}*8 /R) + RTT)} \quad (4)$$

Where
R = user specified rate or line rate (Mbps)
p = packet error rate
File size in megabytes
RTT = round trip time (second)

Figure 2 shows the theoretical throughput for a pure-rate-based and TCP protocols. In this figure, data is acquired with the file size held at a constant at 100 MB while the delay (RTT) is varied from 10 to 500 ms. For the theoretical calculations, a transmission rate of 100 Mbps was used as that was the available bandwidth of our link. In addition, a packet size of 1024 bytes was used. Thus, Figure 2 illustrates the theoretical upper bound for a rate-based protocol.

Figure 2 also shows the theoretical throughput of TCP for three delays. Notice that current deployments of TCP – in particular Reno and SACK – are dramatically effected by errors on the link. In addition, performance drops off rapidly in high-delay environments. Also, notice that a rate-base protocol will far outperform current deployments of TCP in error-prone, high-delay environments.

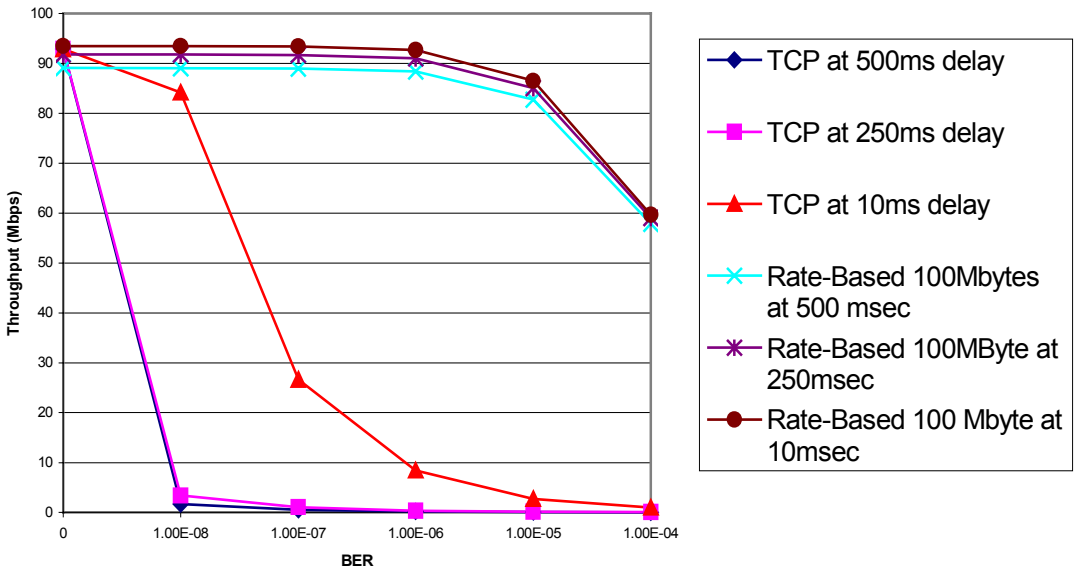


Figure 2. Theoretical Throughputs of TCP and Rate-Based Protocols (packet size =1024 bytes)

7 SYSTEM/PROTOCOL IMPLEMENTATION PROBLEMS

This section contains a chronological summary of the various problems encountered and their associated resolutions.

7.1 NetBSD

Originally, the NetBSD machines were to be used as the receiver and sender systems, but obtaining a valid TCP baseline with these systems was not possible. Some out-of-order packets were observed being sent from the sender during delays of 10 ms. Packet retransmissions were observed when delays were 250ms or higher. These factors led to very poor throughput performance in any type of delayed environment.

In an attempt to improve performance of NetBSD TCP, parameters such as tcp_sendspace, tcp_recvspace, sb_max, and nmbcluster, were increased. In addition, the network interface cards on the receiver and sender machines were changed from 100 Mbps to 10 Mbps. The throughput continued to be relatively low at high delays. Unable to resolve these performance conflicts in a timely manner, two Sun Solaris machines were selected to be used as the sender and receiver systems since preliminary testing showed no such performance problems as compared to the NetBSD machines.

7.2 TCP/SCPS Tuning Problems

As indicated in the document "TCP Tuning Guide for Distributed Application on Wide Area Networks" [17], the optimal sender/receiver buffer size is calculated to be twice the Bandwidth Delay Product (BDP). Therefore, our initial tests were tuned with the sender and receiver buffers set between one and two times the BDP.

Using the Sun machines, baseline tests were conducted for TCP-SACK, transferring 100 MB files in a zero BER environment with 250 and 500 ms RTT delays. During these tests, retransmitted packets occurred during the transfers when the receiver's buffer was set greater than the BDP and when the tests were executed back-to-back.

In some cases, the retransmissions occurred in conjunction with the original segments missing in the receiver side's tcpdump output, indicating that the packet was indeed lost. Most of the time, the receiver still generated the SACKs to trigger the retransmissions, even when the original packets appeared on the tracing machine interface of receiving side [Figure 3]. It is suspected that the receiving machines were unable to keep up with the high speed transfers and that packets were dropped at the receiver interface or processed incorrectly from the interface up to the transport layer.

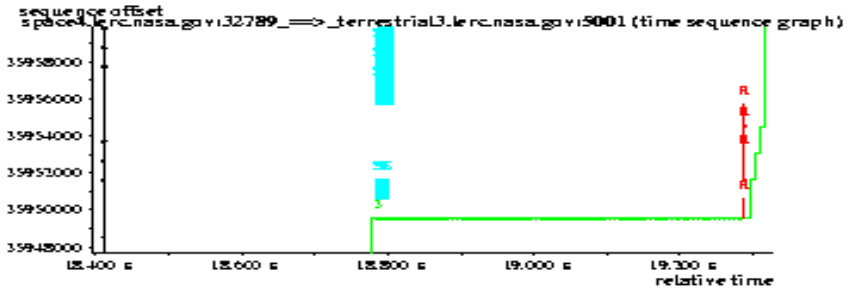


Figure 3. Unnecessary SACKs

While tuning the SCPS-TP protocol at the line rate of 100 Mbps, unexpected retransmitted packets were also observed. This hindered the protocols' abilities to operate at optimal levels. An exception to this observation was the SCPS-VJ protocol, which was able to support the 100 Mbps line rate speed without retransmissions.

7.3 Tuning Method

In order to obtain the best performance of TCP and SCPS-TP tests, 15 tests of 100 MB files were transmitted at the three delays. The values of the receiver buffer started at double the BDP, decreasing to about 90% of the BDP. The receiver buffer that gave the highest average throughput and the lowest standard deviation, was then selected.

As a result of the previously described tuning method, the following values of the receiver buffer size were used for both the single flow TCP and SCPS-TP tests:

Delays (ms)	Receiver buffer (MB)
500	6.25, 6.2, 6.1, 6.0, 5.9, 5.85, 5.8, 5.7
250	6.25, 3.125, 3.1, 3.0,2.9, 2.85, 2.8, 2.7
10	0.250, 0.125, 0.12, 0.118, 0.116, 0.114

The optimal receiver buffer sizes and setting of other parameters for each individual testing set is provided in theTesting Results Section and in Appendix A through F.

7.4 SCPS-TP Vegas

While tuning for the SCPS-Vegas tests at a 500 ms delay, for 100 Mbps transmission rate and a zero BER, it was occasionally observed that some transfers had retransmitted packets. This occurred when the receiver's buffers were set between 90% and 100% of the BDP. This phenomenon of retransmitted packets was observed until the transmission rate was lowered to 60 Mbps, which was then determined to offer the best performance for the SCPS-Vegas tests.

7.5 SCPS-TP Pure Rate-Based

In the MITRE implementation of SCPS-TP Pure Rate, the sending rate in the SCPS-Pure Rate testing depends on the set transmission rate and the receiver window. In our tests, the setting of the receiver window sizes under the three delays was not a factor that limited the sending rate. In both the SCPS-TP and Pure Rate tests, the highest average throughput was always about 44 Mbps with the transmission rate set to 45 Mbps or greater. In addition, SCPS-TP Pure Rate did not have an optimum performance at a setting of 100 Mbps. Empirical results indicated that 80 Mbps was the maximum setting under which the MITRE implementation would operate properly. Therefore, a sending rate of 80 Mbps was used rather than 100 Mbps.

7.6 SCPS-TP Van Jacobson

While running the SCPS-VJ baseline tests, transfers using the Sun machines slowed down without taking any losses or having out-of-order packets [Figure 4]. When using the NetBSD machines as the ttcp sender and receiver, a throughput of around 70 Mbps could be achieved while transferring 100 MB files at a 10 ms delay over an error-free link. This problem may be related to the behavior of the Solaris operating system as it pertains to the SCPS protocol running outside kernel, but can only be substantiated with further investigation.

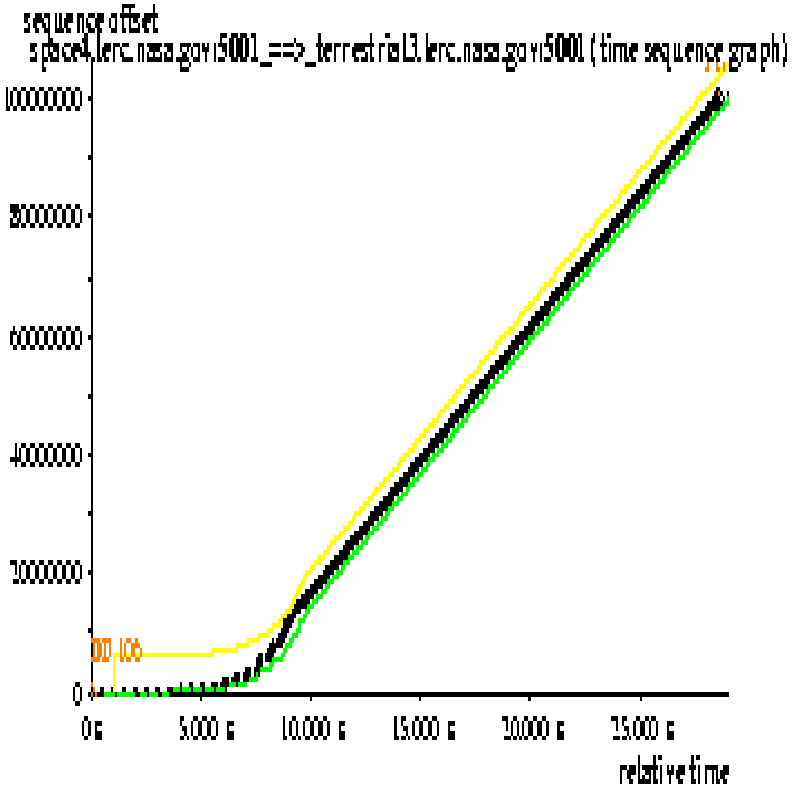


Figure 4. Questionable Throughput Decrease

7.7 Multiple Flow Testing

Initially, it was attempted to use a single pair of Sun sender/receiver machines for multiple flow testing. In this method, several scps_ttcp processes were started between the sender and receiver, with each process using the same physical interface but directed to different a port, by use of a dynamic port option (scps_ttcp –p).

Unlike regular TCP using the ttcp program, the unmodified SCPS_RI code would always send a packet out of port 5001 from the sender. This occurred regardless of the port number indicated with the –p option. The regular TCP ttcp program would send the packet out of unique high port numbers with each invocation of the ttcp code. As a result, three copies of the scps_ttcp code were compiled, where each code had a different port number hard-coded into the compilation.

Unfortunately, further multiple flow testing with these hard-coded port changes showed that load sharing of the three scps_ttcp receive and send processes on the one pair of sender and receiver machines took precedence, thus giving undesirable results for this emulation. Regardless of whether one or two of the three flows were still in transmit while the previous flow had completed, all three transmissions would result in the same average throughput, which would equate to 1/3 of the available bandwidth. This occurred even if the first transmission had ended a time before the second and third, or if the third was still transmitting while the first and second flows had completed. With fewer

packets on the link to cause congestion, it was expected for the remaining executing flows to transmit more packets, but this was not the case on this single pair of machines. The eventual multiple flow testing that was used for the final test results was then performed on three pairs of separate Sun sender/receiver machines, without the hard-coded port changes in the SCPS_RI code.

7.8 MDP

Initially setting the transmission rate above 4Mbps caused the MDP receiver to crash when file sizes of 10MB or greater were being sent. As a result, the MDP source code was obtained and recompiled with a MAXIMUM_RX_BUFFER set to 10MB. After recompiling the MDP source, the MDP sender was able to send at rates up to 100Mbps without crashing the receiver, however the goodput performance was abysmal.

Efforts to increase MDP's goodput included eliminating disk I/O by using a memory file system. When performance did not increase measurably, tcpdump files were examined to determine if the sender was sending at the user-specified rate. Further examination of the problem revealed the receiver's resources were being overrun by the MDP sender. Tests were then conducted under ideal delay and BER conditions, which revealed that the maximum performance attained by MDP on our test systems was from a user-specified rate of 40Mbps. Further illustration of the MDP throughput can be found in Figure 5.

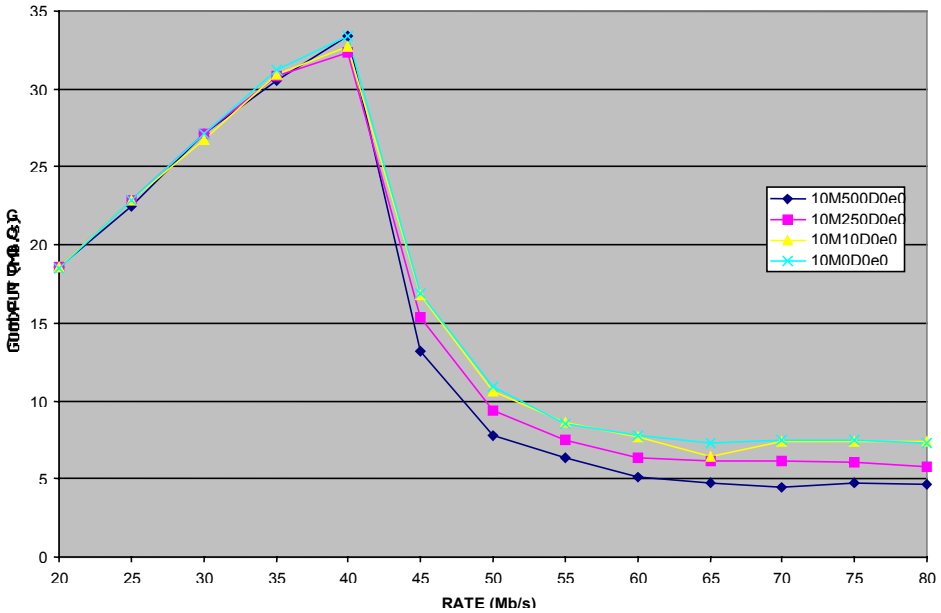


Figure 5 RATE vs GOODPUT (BER 0e0 10MB Files)

For 100 MB transfers, MDP would often hang and would not compete the transfer. For these large files, MDP appeared to have its packet sequence numbers wrap around. We hypothesize that this may be the cause of the problem. This phenomenon requires further investigation.

8 TESTING PHILOSOPHY

Our goal was to evaluate these protocols in a space-based environment (long-delay, error-prone links). However, for completeness, tests were conducted using RTT delays of 10 to 500 ms over the full range of BERs (0e0 to 1E-4). It should be noted that a 10 ms delay is not indicative of a space-based environment, but a LAN. Thus, having a real-world situation where there are BERs of 1E-7 or higher with 10 ms delays, is unlikely unless one is considering a wireless LAN.

Although initial testing of the TCP and SCPS protocols under BSD yielded better performance for some configurations (see Section 10 Recommendations for Further Testing), testing under Solaris produced results which more closely matched the theoretical characteristics of the test protocols. For these reasons, Solaris was selected as the test platform for all protocols.

During the initial investigations and baseline tests, it was noted that neither TCP nor SCPS were performing with a great degree of consistency when compared to theory or protocol specifications. Results obtained from testing would range from optimal (theoretical) to sub-optimal. It was concluded that data yielded from all testing should be deemed valid, and should not be discarded. Instead, inconsistencies were thoroughly examined to determine if the sub-optimal results were due to protocol implementation, test strategy, or system (e.g. Hardware, OS) implementation. Sub-optimal results that were due to testing problems or known protocol bugs, were re-conducted with the issue(s) addressed. In most cases, the most logical explanation for inconsistent results pointed to system implementation, which falls outside of the scope of this group. As a result of having to deal with inconsistencies that fall outside of our control, our testing philosophy evolved to the following:

- Optimize and baseline protocol on an error-free link for each BDP
- Perform a minimum of 20 to 30 individual transfers for each series
- Record all measurements (not just optimal runs)
- Capture and save all SYN/FIN packet traces and some dump files for each test series.
- Examine results as an entire series with inconsistencies due to controllable circumstances resulting in the re-conducting of the entire series.

9 TESTING RESULTS

For each series of tests, 10 to 30 transfers were conducted to obtain the average transfer time and standard deviation. Adhering to our testing philosophy, average results with a large standard deviation were examined and re-conducted if necessary. Transfer time and subsequently throughput results of each individual test were established dependant upon the protocol. In the case of TCP and SCPS, these statistics were determined from the tcpdump timestamp difference between the initial SYN and the final FIN captured on the sender side. Because MDP and MFTP do not use TCP's three-way handshake and tear down, the transfer start was determined to be the time between the tcpdump timestamp of the first full length data packet seen from the sender and the timestamp of the first empty non-acknowledgement seen from the receiver.

The results attained from testing can be visualized in a variety of ways. In an effort to be concise, the reader is being provided graphs depicting the overall performance of each protocol in the form of average throughput within controlled BER conditions. Where applicable, transfers of different delays and file sizes are plotted to illustrate observations made, or conclusions drawn, as a result of testing. Should the reader wish to make further comparisons, raw data from all tests conducted is available in the appendix.

9.1 Single Flow Congestion-friendly Protocols

9.1.1 Configurations

9.1.1.1 TCP-SACK

For the TCP-SACK test, the default values set on the Sun Solaris 7 kernel were used for most of the TCP/IP parameters, except for the following:

tcp_sack_ permitted 2
tcp_wscale_always 1
tcp_max_buf 16777216
tcp_*cwnd*_max 16777216
tcp_tstamp_if_wscale 1
tcp_tstamp_always 1

The definition of the above Sun TCP parameters can be found at http://docs.sun.com/

9.1.1.2 SCPS with Van Jacobson Congestion Control and Acknowledge Every Other Packet (SCPS-VJ)

For the SCPS-VJ test, the default Delayed ACK timer was changed from 200 ms to 50 ms in order to be consistent with the TCP tests. The other SCPS-TP parameters were left unchanged. Both the SCPS rate option ("-R " option) was tried at 80 Mbps and 100 Mbps. Since average throughput appeared a little higher at 100 Mbps, the SCPS rate of 100 Mbps was chosen for this testing set.

For both the TCP-SACK and SCPS-VJ, tests for the 100 MB file at BER 1e-5 and the other three file sizes (10 MB, 1 MB, 100 KB) at BER 1e-4 were not conducted, as the average throughput of the biggest file size and the other file size transfers were already low at 1e-6 BER and at 1e-5 BER, respectively. "Too low throughput" was understood to be less than 0.5% of the baseline rate (eg. less than 0.5 Mbps for 100 MB under both TCP and SCPS_VJ at a delay of 500 ms), causing the file transfer tests to run in an excessive amount of time.

Appendix A and B show the sample command, the optimum receiver buffer size values, average throughput and standard deviation of the TCP-SACK and SCPS-VJ tests.

9.1.1.3 SCPS-Vegas-Congestion and SCPS-Vegas-Corruption

The only difference between SCPS-Vegas-Congestion and SCPS-Vegas-Corruption is SCPS-Vegas-Congestion reduces *cwnd* by half in response to a dropped packet, while the *cwnd* in SCPS-Vegas-Corruption remained unchanged.

SCPS-RI version 1.1.62 was used for the SCPS-Vegas-Congestion tests, whereas version 1.1.66 was used in the SCPS-Vegas-Corruption tests. The SCPS version 1.1.66 is similar to version 1.1.62; however, version 1.1.66 had an additional switch option for the Vegas methods allowing either the original slow-start or the modified slow-start mechanism as described in Section 4.2.2.

Due to a time constraint of the testing, only the Vegas tests of 10 and 100 MB files were performed, with the number of transfers in each test series reduced. . The optimum rate setting for both Vegas tests were 60 Mbps at 500 ms delays, and 80 Mbps at 250 ms and 10 ms delays.

Appendix E and F show the sample command, the optimum receiving buffer size, average throughput, and standard deviation of the Vegas-Congestion and Vegas-Corruption tests.

9.1.2 Comparisons: SCPS-Vegas-Congestion, SCPS-Vegas-Corruption, TCP-SACK, and SCPS-VJ

Figure 6 shows the average throughput vs. BER for 10 MB and 100 MB file sizes over a link with a 500 ms delay. Figure 7 shows the average throughput vs BER when transmitting a 10 MB over various delays. In general, the overall characteristics of the three congestion-friendly protocols were similar. Both TCP-SACK and SCPS-VJ use the Van Jacobson Congestion Control algorithm; thus, the overall characteristics of the SCPS-VJ are similar to that of TCP-SACK. SCPS-Vegas-Congestion performs slightly better then TCP-SACK. Notice that, in all cases, larger files have a better throughput at zero BER than smaller files. This is because *slow start* has a less effect as files get larger. Also, notice that smaller files have better throughput at higher BERs than larger files. For TCP-SACK and SCPS-VJ, this is due to the additive increase, multiplicative decrease congestion control algorithms.

Since the *cwnd* in the SCPS-Vegas-Congestion transfers is cut in half each time an error occurs, their average throughput declined faster than that of SCPS-Vegas-Corruption transfers when errors were inserted into the network. Furthermore, SCPS-Vegas-Corruption performed better than all congestion-friendly protocols at higher BER, but still did not perform as well as a pure rate-based protocol as congestion control is still being performed.

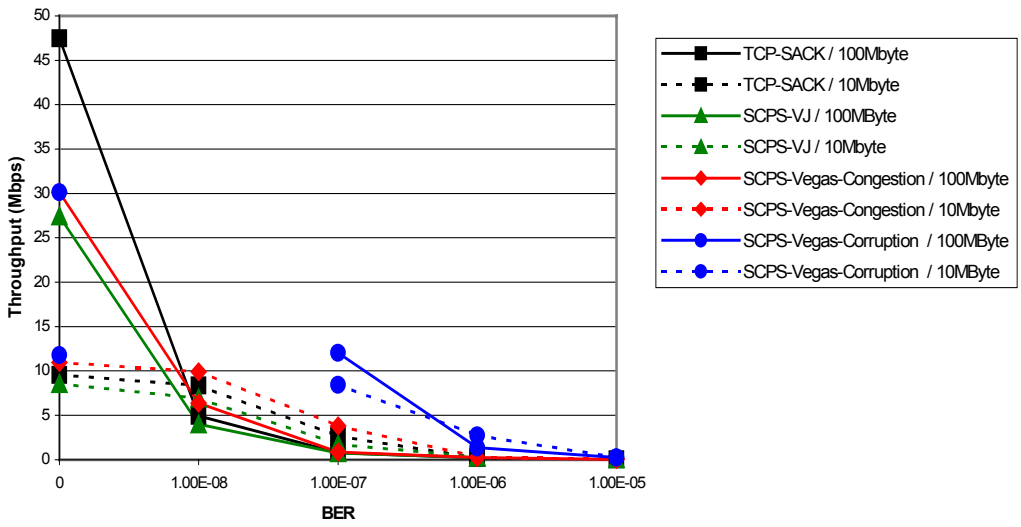


Figure 6. Congestion-Friendly Protocol Performance (500 ms Delay)

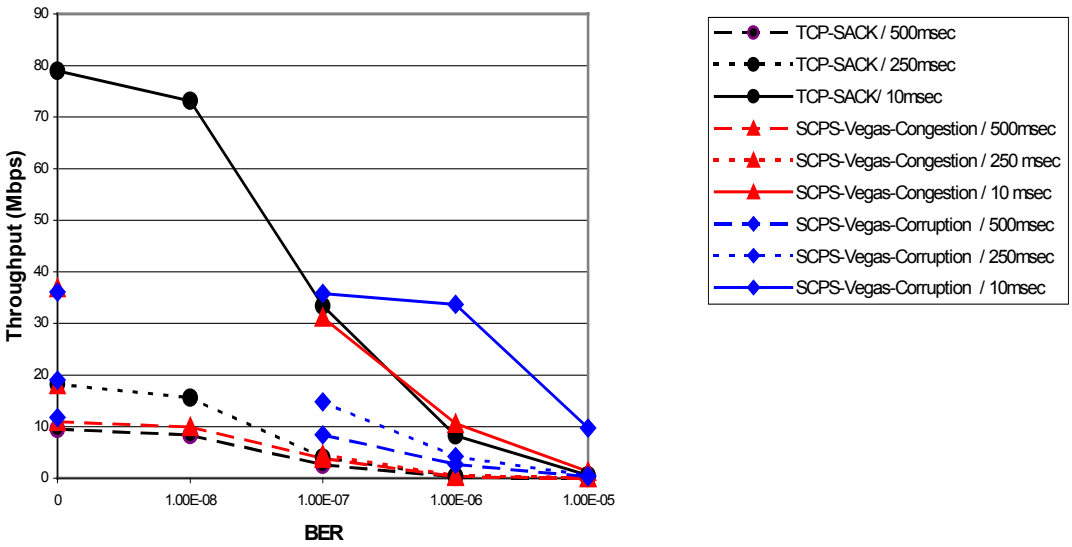


Figure 7. Congestion-Friendly Protocol Performance (10 MB File Transfer)

9.2 Single Flow Rate-Based Protocols

Since there is no congestion control mechanism involved in these tests, Pure-Rate Control tests can be used only when there is no congestion occurring in the network. For this reason, tests were conducted as a single flow, one transfer at a time. Testing in a multiple flow environment would be inappropriate.

9.2.1 Configurations

9.2.1.1 SCPS Pure-Rate Control with Acknowledge Every Other Packet (SCPS-Pure-Rate-2) and Pure-Rate Control with Strictly Delayed Acknowledgements (SCPS-Pure-Rate-F0)

For the SCPS-Pure-Rate-F2 tests (Pure-Rate Control with Acknowledge Every Other Packet), SCPS-RI version 1.1.51 was used with the Delayed ACK Timer changed to 50 ms. The optimum rate for this test set was determined to be 80 Mbps as determined through the previously described tuning tests.

However, while tuning for the SCPS-Pure-Rate-F0 tests using the SCPS-RI 1.1.51 version, it was noticed that the acknowledgements were not being sent back every 200 ms as defined by the default Delayed ACK Timer. Instead, the acknowledgements were being sent sometimes much later than 200 ms. The shortest time for the ACKs returning back to the sender was 200 ms, and the longest time could be 1 to 2 seconds. This reduced rate of returning of ACKs led to a receiver window-limiting problem because the receiver window was not getting updated fast enough. Therefore, a new version of the SCPS-RI was used, version 1.1.62, which did send the ACKs correctly per the defined Delayed ACK Timer. As a result, the Delayed ACK Timer for the SCPS-Pure-Rate-F0 tests was left unchanged at the default of 200 ms using the new SCPS-RI 1.1.62. Like the SCPS-Pure-Rate-F2 tests, 80 Mbps was also determined to be the optimum rate for the SCPS-Pure-Rate-F0 test.

Appendix C and D show the sample command, the optimum receiving buffer size, average throughput, and standard deviation of the SCPS-Pure-Rate-F2 and SCPS-Pure-Rate-F0 tests respectively.

9.2.1.2 MFTP

Systems involved in the MFTP transfers were configured at the application's maximum rate of 50Mbps. In addition, the unicast address was specified for both systems. The MFTP server was set transmit the file only once, but to transmit repair data in answer to client NACKS for up to one hundred times the initial transfer time, or until client signals completion with an empty NACK.

9.2.1.3 MDP

Systems involved in the MDP were configured to use their unicast address at an optimal rate of 40 Mbps . The server was set to send data blocks without parity (forward error correction), and to retransmit its payload for an indefinite number of passes (server resends repair data until an empty NACK is received). The receiver was configured to archive the data and to refrain from any post processing (the default was to open the data

in a web browser) as this was the only way to get the MDP receiver to log a file transfer completion.

9.2.2 Comparisons: SCPS-Pure-Rate-F2, SCPS-Pure-Rate-F0, MDP, and MFTP

Figure 8 shows the performance of the various rate-based protocols for a 500 ms RTT delay. None of the rate-based protocols tested came close to meeting the theoretical throughput when large files were transmitted. This may be due to the capabilities of our machines along with implementation issues. Although the experimental throughput of all rate-based were lower than the theory, the 10 MB MFTP curves at the three delays matched closest to the shape of the theoretical curves. Under all implemented BER and DELAY conditions, MDP performed well up to approximately 35 Mbps and slightly better than both of the SCPS-TP Pure Rate protocols (see Appendix G). Figure 5 illustrates an MDP performance fall off, possibly due to receiver processing problems at higher-transmission rate settings. In fact, for 100 MB files, MDP would not complete the transfer as the receiver would choke (possibly due to the previously discussed packet number wrap around phenomena). SCPS rate-based protocol also fell off faster than predicted at the high BERs and did not perform anywhere near theoretical throughput, even at low BERs. As with MDP, this may have been due to the receiver becoming overwhelmed and unable to keep pace with the sender – particularly at higher BER and high delay.

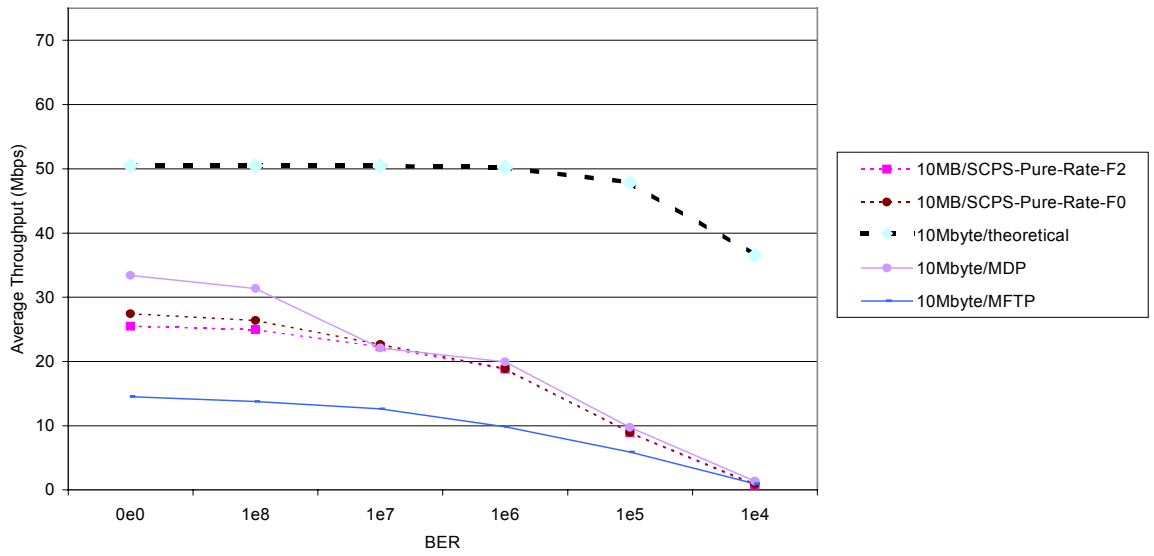


Figure 8. Rate-based File Transfers with 500ms Delay

Note, in Figure 9, the SCPS-Pure-Rate-F2 and -F0 curves at a 10 ms delay closely follow the shape of the corresponding theoretical curves as compared to these curves at 500 ms delays. This leads us to believe this may be a memory management problem. In addition, an in-kernel implementation of SCPS may improve performance [18, 19].

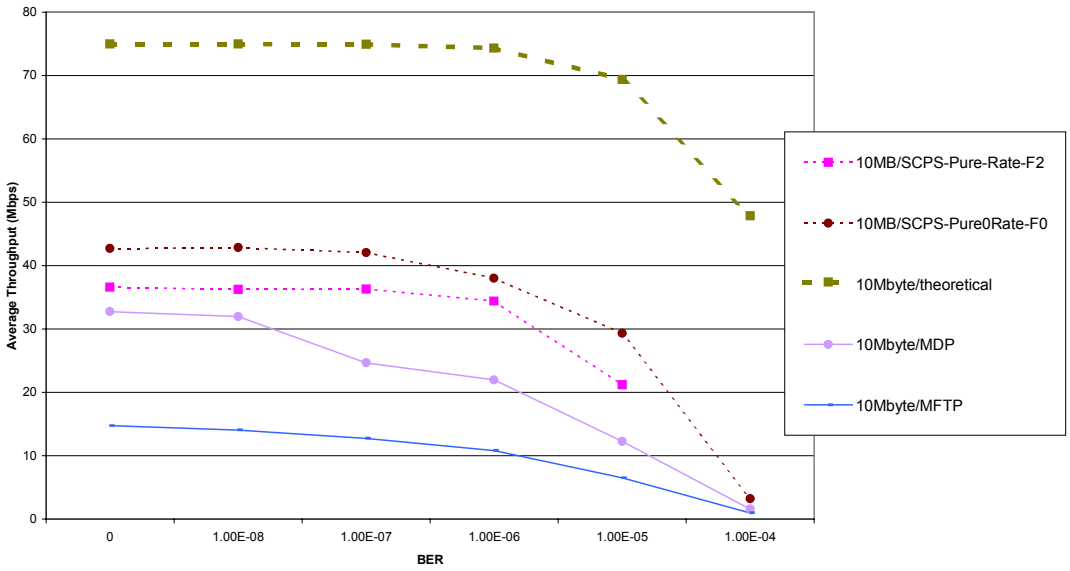


Figure 9. Rate-Based Average Throughput vs. BER for 10 ms Delay

9.3 Multiple Flow Results

The multiple flow tests were designed to exercise the congestion control properties of the congestion-friendly protocols. In the single flow tests, an individual flow could utilize the entire bandwidth. For the multiple flow tests, three pairs of flows competed for the available bandwidth. For these tests, the bandwidth was set to 15 Mbps at the ATM interfaces of the routers. Time constraints allowed only a limited subset of tests to be performed with BERs of 0, 1e7, and 1e5, a RTT delay of 500 ms, and a single file size of 50 MB. In addition, SCPS-Vegas-Corruption was not tested in this environment, as it would have been a misapplication of the protocol.

Similar to the single flow tests, SCPS-Vegas-Congestion performed slightly better than TCP-SACK, which performed slightly better than SCPS-VJ at zero and moderate BERs, with RTT delay of 500 ms. [Figure 10]. At a BER of 1e-7 and 1e-5, the throughput of each pair in multiple flow tests had almost the same performance as in the single flow tests under these same error conditions. This was because the packet loss due to errors dictates the performance rather than actual congestion. Notice that the total average throughput can exceed the network capacity. This is due to the random offset start times for the three flows, where the flows are all started at random intervals, causing individual flows to transfer and complete during different usages of the available bandwidth. The total average throughput of any one set of tests can exceed the 15 Mbps, particularly if the first and last transfers do not overlap by much.

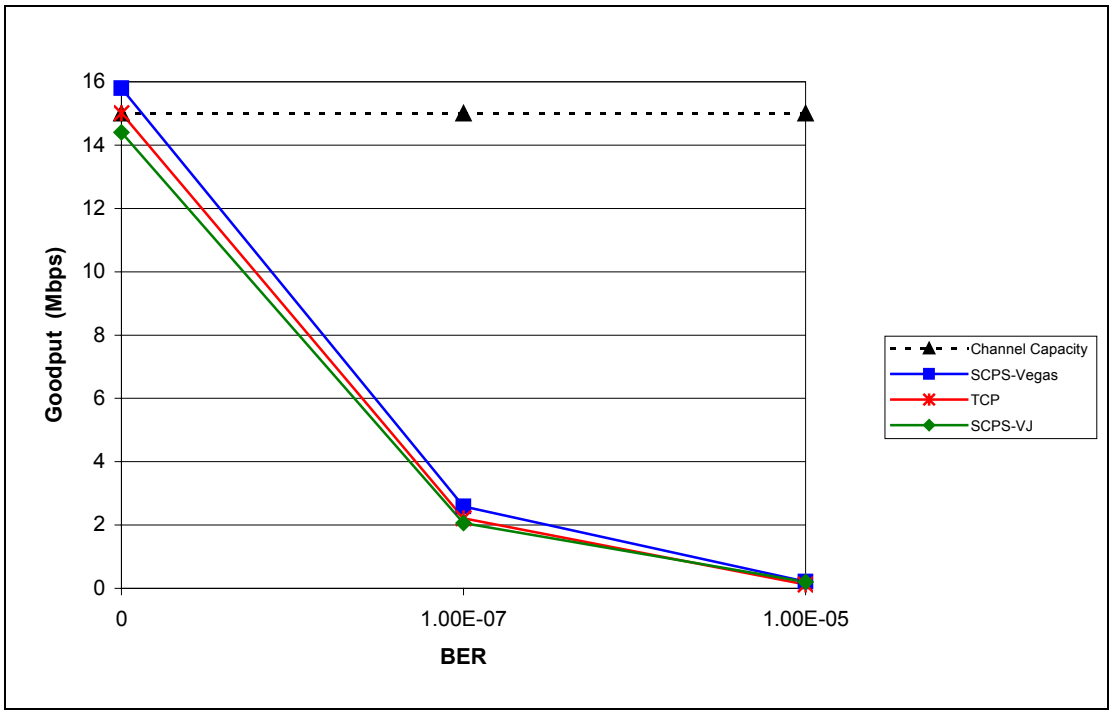


Figure 10. Multiple Flow Goodput vs BER

Figure 11 shows the throughput of each TCP-SACK flow in individual runs for all 30 tests. Tests reflect each transmitter/receiver pair of machines at zero BER. Similar results occurred for the SCPS-VJ and SCPS-Vegas-Congestion (see appendix for all three graphs). A detailed analysis of the individual data runs shows that no flow had a throughput below 3 Mbps in all 30 test runs of SCPS-Vegas-Congestion. However, in the TCP-SACK and SCPS-VJ tests, there are 20 out of 30 test runs having a throughput below 3 Mbps. While the TCP-SACK and SCPS-VJ tests have a throughput rate of 2.3 through 7.9 Mbps and 1.9 through 7.5 Mbps, respectively, the minimum and maximum throughput of the flows in the SCPS-Vegas-Congestion tests are 3.2 and 9.5 Mbps.

Although not shown in the graphics, it is important to note that at a BER of zero, the percentage of retransmitted packets in SCPS-Vegas-Corruption was almost none as compared to 0.3% and 0.16% under TCP-SACK and SCPS-VJ. The improvement occurs because, under SCPS-Vegas-Congestion Control, the *cwnd* is monitored between the *alpha* and *beta* thresholds.

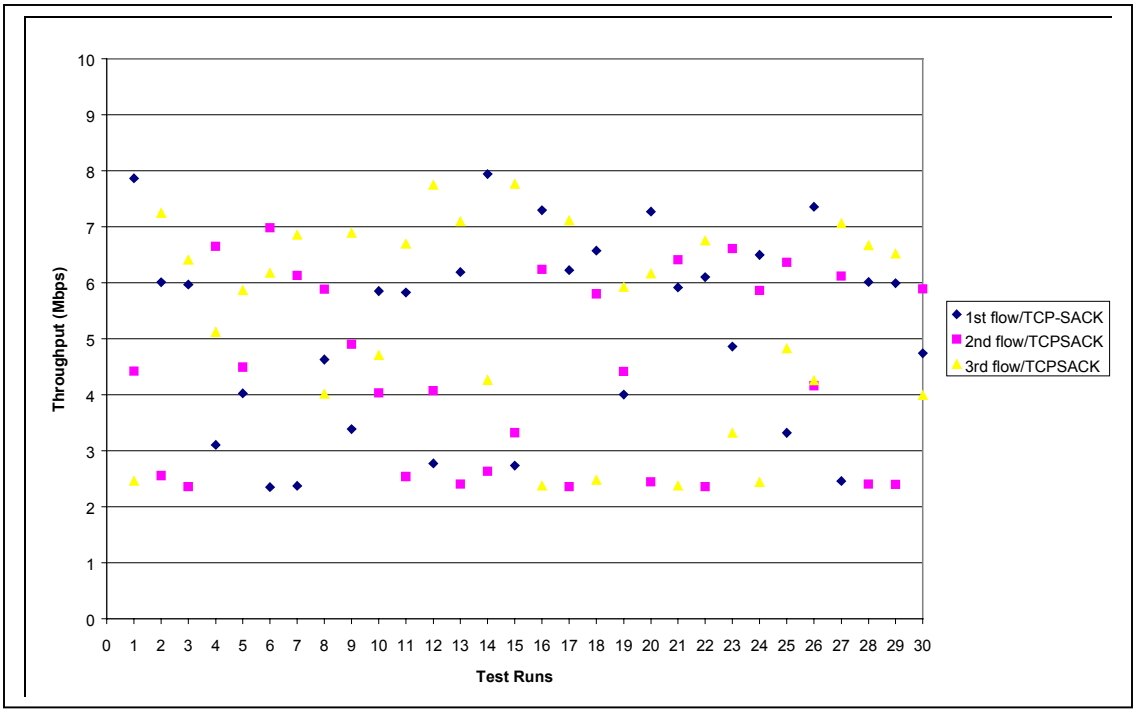


Figure 11. Individual Multiple Flow Transfers

10 RECOMMENDATIONS FOR FURTHER TESTING

During the preliminary testing, 100 MB files were transferred at 100 Mbps with a 10 ms delay, using an error-free link between two NetBSD machines. The throughput of these SCPS-VJ and TCP-SACK tests were around 70 Mbps and 79 Mbps, respectively. However, while running the same test for SCPS-VJ and SCPS-Vegas-Congestion using Solaris machines as sender and receiver, it was noticed that the maximum average throughput was about 38 Mbps, which was much lower as compared to 87 Mbps for TCP-SACK on Solaris. Furthermore, with the Solaris sender and receiver under the 100 Mbps link capacity, none of the SCPS-Pure-Rate transfers could achieve a throughput higher than 44 Mbps despite the SCPS setting rate. This low throughput of the SCPS transfers may possibly be caused by the Solaris operating system, which may have some effect on processes running outside the kernel, such as SCPS-TP. This needs further investigation.

Also, with the zero BER and 500 ms delay link, the throughput of a 100 MB file transfer under SCPS-Pure-Rate-F2 using a NetBSD sender and a Solaris receiver does not respond correctly to the SCPS setting rate when the rate is set lower than 40 Mbps. The throughput was 14 Mbps when the SCPS rate option was set at 30 Mbps, even though there were no dropped packets in the transfer. On the other hand, the same transfer between two Solaris machines gave a throughput of 27 Mbps (with the SCPS rate set to 30 Mbps). This needs further investigation.

All rate-based protocols need further testing on faster machines and for different operating systems to determine if that will improve performance or if these protocols

need further development. For the commercial rate-based protocols tested, this poor performance may be due to the algorithms and coding being optimized for multicast operation. Therefore an investigation of SCPS performance, under different operating systems, using faster machines are suggested. Also, using an in-kernel SCPS-TP implementation may lead to better performance of SCPS in an error-free environment.

The SCPS rate-based protocols utilize the TCP header and appear as sender only modifications, the protocol is advertising TCP as the protocol number. However, SCPS-TP Pure Rate is not performing congestion control. The authors suggest that, for such operation, SCPS-TP should advertise a different protocol number to ease quality-of-service provisioning. Failure to do so may result in SCPS rate-base flows dominated shared links or being identified as rogue sources.

SCPS-Vegas should be further investigated for operation in mobile environments, and for performance on intermittent links as the Vegas algorithm has some known problems [20, 21].

- Vegas uses an estimate of the propagation delay, and base RTT to adjust its window size. Thus, it is very important for a TCP Vegas connection to have an accurate estimate of this quantity. Rerouting a path may change the propagation delay of the connection, and this could result in a substantial decrease in throughput.

- Each TCP Vegas connection attempts to keep a few packets in the network. When the estimation of the propagation delay is off, this could lead the connections to inadvertently keep many more packets in the network, causing a persistent congestion.

- The Vegas congestion avoidance algorithm intentionally lowers its transmission rate under heavy congestion. Thus, in head-to-head transfers, TCP-Reno steals bandwidth from Vegas. This is one possible reason why Vegas has not seen wide deployment in the Internet.

11 CONCLUSIONS

We have studied the effect of delay and BER on the performance of congestion-friendly and rate-based protocols in uncongested and limited congested emulated space links. The results correlate well with other testing of SCPS-TP and TCP.

- In a space-based environment, the single stream and multi-stream test results clearly illustrate that the SCPS-Vegas enhancements to TCP provide measurable performance improvements over the TCP SACK implementation tested. The value of these performance increases is subjective and would need to be judged on a mission by mission basis.
- Very small transactions such as command and control should see little difference in performance for TCP or any variant of SCPS-TP or a rate-based protocol.
- In extremely error-prone environments with high RTT delays, a rate-based protocol is advisable if you properly engineer the network. However, one must beware of using

rate-based protocols on shared networks unless you can reserve bandwidth. In addition, rate-based protocols may be applicable for any environment where bandwidth reservation is practical and available.

- The deployment of an in-kernel protocol may be more desirable than the deployment of an application level protocol, for more efficient use of resources and performance issues. However, one may also argue, that it is far easier to maintain a protocol at the application level than within a kernel.
- Even with equal performance, the SCPS rate-based protocol may more desirable to implement than other rate-based protocols such as the multicast Dissemination Protocol , as SCPS is capable of requiring only sending-side only modification.
- The existing standard transport protocol[4] and capabilities (drawn from a variety of communities) appear to satisfy all known mission needs; however, the space community should maintain as awareness of current and future TCP research. New TCP research may dramatically improve TCP operation for near-*planetary* environments. Some pertinent areas include Stream Control Transmission Protocol (SCTP), TCP Pacing with Packet Pair Probing, TCP Westwood, and TCP Explicit Transport Error Notification (ETEN).

12 ACKNOWLEDGEMENTS

The authors would like to thank the following people who either offered technical support on protocol problems, offered their expertise in conducting tests and analyzing results, or provided hardware resources needed by the research team:

- Keith Scott
- Pat Feighery
- Eric Travis
- Charlie Younghusband
- Adrian Hooke
- Brian Adamson
- Mark Allman, who is the greatest task lead of all time (in case he's actually reading this :^)
- Jim Griner , who's best chance of flying solo, will be to flap his arms.

13 REFERENCES

[1] Darrell E. Ernst, R. Durst, "Space Communications Protocol Standards (SCPS) Bent-Pipe Experiment Report", SCPS D71.51-Y-1, May 1996

[2] S. Horan and R. Wang, "Comparison of File Transfer Using SCPS FP and TCP/IP FTP over a Simulated Satellite Channel," Proc. International Telemetering Conference, 34, p. 87-95, October 1999.

[3] J. Muhonen, R. Durst, "Space Communications Protocol Standards (SCPS) FY97 DOD Test Report", February 1998

[4] SCPS-Network Protocol was not evaluated in this study due to lack of hardware and software implementations. All routing was performed over IPv4, which is deployed throughout the Internet. Note that SCPS-NP will not accommodate the National Security Agency's High Assurance Internet Protocol Interoperability Specification (HAIPIS); thus, use of SCPS-NP for secure government applications may be problematic.

[4] E. Edwards, C. Younghusband, C. Belise J. Peng, M. Phisel, and L. Hartman, "An Independent Implementation of SCPS-TP with Simulation Extrapolation for Complex Networks", AIAA International Communications and Satellite Systems Conference and Exhibit 12-15 May, 2002 Montreal, Quebec, CA, paper1912

[5] W. Stevens, TCP/IP Illustrated Vol.1, Chapter 20, Addison-Wesley Publishing Company, 1994

[6] W. Stevens, TCP/IP Illustrated Vol.1, Chapter 21, Addison-Wesley Publishing Company, 1994

[7] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, IETF, October 1996

[8] R. Durst, G. Miller, and E.Travis, "TCP Extensions for Space Communications", http://www.scps.org, May 1998

[9] R. Fox, "TCP Big Windows and NACK Options", RFC 1106, IETF, June 1989

[10] L. Brakmo, S. O'Malley, and L. Peterson, " TCP Vegas: New Techniques for Congestion Avoidance", Proceedings of ACM SIGCOMM '94, London, U.K, October 1994.

[11] J. Macker, W. Dang, "The Multicast Dissemination Protocol (MDP) Version 1 Framework", (search: mdpv1.ps), Naval Research Laboratory Technical White Paper, April 1996

[12] W. Dang, "Reliable Multicast Transfer in the Multicast Domain", (search: imm_overview.ps), August 1993

[13] J. Macker, B. Adamson, "The MDP Protocol Specification Version 1.6", (search: DraftMdpSpec-1.6.txt), Naval Research Laboratory, October 1999

[14] K. Miller, K. Robertson, A. Tweedly, M. White, "Starburst Multicast File Transfer Protocol (MFTP) Specification", (search: draft-miller-mftp-spec-03.txt), IETF DRAFT, April 1998

[15] S. Mainger, R. Dimond, "Evaluation of GIBN TRANS-Atlantic Demonstration", NASA-GRC, April 2000

[16] M. Mathis, " The Macroscopic Behavior of the Congestion Avoidance Algorithm", Computer Communications Review, Volume 27, Number 3, July 1997.

[17] "TCP Tuning Guide for Distributed Application on Wide Area Networks", http://www-didc.lbl.gov/tcp-wan.html

[18] http://www.xiphos.ca, July 2002

[19] http://www.skipware.com, July 2002

[20] "Space Communications Protocol Specification (SCPS) – Transport Protocol (SCPS-TP)", CCSDS 714.0-B-1, Blue Book, May 1999

[21] J. Mo, R. J. La, V. Anantharam, and J.Walrand, "Analysis and Comparison of TCP Reno and Vegas", Infocom 1999

Appendix A: TCP-SACK

Sample Session for 500ms Delay
Receiver ttcp –b 5700000 –r –s
Sender ttcp –l 1024 –t destination < input file

Buffer Size
5.7 Mbytes for 500 ms Delay
2.85 Mbytes for 250 ms Delay
250 Kbytes for 10ms Delay

Average Throughput

Delay 500	Tput	(Mbps)			
File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	47.4834	4.906531	0.777052	0.216365	N/A
10MB	9.54826	8.349969	2.62682	0.68263	0.058365
1MB	1.35358	1.34096	1.042389	0.38253	0.04743
100KB	0.20599	0.20433	0.198737	0.16356	0.059725

Delay 250	Tput	(Mbps)			
File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	63.485866	9.499695	1.439371	0.426265	
10MB	18.196483	15.631853	3.961263	0.4411088	0.0850289
1MB	2.599809	2.573242	2.222325	0.76744	0.092774
100KB	0.393329	0.407028	0.401099	0.333063	0.1113048

Delay 10	Tput	(Mbps)			
File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	87.479128	77.952514	29.868459	8.226708	0.616309
10MB	78.9544632	73.172567	33.412264	8.268011	0.6126601
1MB	37.390096	36.902548	34.869892	12.324005	0.482513
100KB	6.698032	6.7909025	6.790415	5.311491	0.4433649

Standard Deviation

Delay 500 ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.839478	4.29734	0.070928	0.037817	N/A
10MB	0.302687	2.232559	2.623635	1.731995	0.01196
1MB	0.06453	0.104001	0.333633	0.294175	0.005681
100KB	0.040612	0.014449	0.024918	0.036164	0.030739

Delay 250 ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.584809	4.943528	0.11151	0.039137	N/A
10MB	0.591234	4.188728	4.427847	0.050411	0.004
1MB	0.103236	0.18669	0.478944	0.564232	0.020954
100KB	0.024339	0.03139	0.33995	0.087011	0.070981

Delay 10 ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.114941	3.842522	1.664462	0.211471	0.072359
10MB	0.634929	9.393886	11.086965	0.99136	0.111251
1MB	4.284046	4.39584	8.495563	8.862179	0.286997
100KB	0.209645	0.380972	0.26802	2.252438	0.850613

Appendix B: SCPS-VJ

Sample Session for 500ms Delay

Receiver

scps_ttcp –b 6000000 –G 1 –F 2 –R 100000000 –f m –r -s

Sender

scps_ttcp –b 6000000 –G 1 –F 2 –R 100000000 –f m –t Destination < input file

Buffer Size / Rate

6Mbytes/100Mbps for 500ms Delay
2.85Mbytes/100Mbps for 250ms Delay
116Kbytes/100Mbps for 10ms Delay

Average Throughput

Delay 500 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	27.421424	4.0032943	0.7384302	0.266285	N/A
10MB	8.550744	6.9033921	1.72777	0.27286	0.073143
1MB	1.254006	1.240654	1.0614931	0.382902	0.074817
100KB	0.226464	0.20307	0.20307	0.176559	0.080338

Delay 250 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	31.421814	6.487708	1.47184288	0.522026	N/A
10MB	15.206519	13.503351	2.45793255	0.525609	0.145188
1MB	2.4908702	2.38995668	1.9172916	0.771067	0.149487
100KB	0.4488258	0.43968877	0.4437284	0.371175	0.1729023

Delay 10ms Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	37.984115	35.949703	25.095862	9.135654	1.517782
10MB	36.682395	35.115305	24.739816	9.269285	1.505615
1MB	27.681829	27.287502	23.45413	9.266045	1.6729085
100KB	7.957629	7.96509	7.783806	6.867243	2.638308

Standard Deviation

Delay 500ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.15572	2.545508	0.04628	0.00425	N/A
10MB	0.055137	2.421036	1.536055	0.017517	0.002763
1MB	0.044958	0.081583	0.277809	0.269686	0.007534
100KB	0.000309	0.064875	0.064875	0.052183	0.032432

Delay 250ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.120869	2.785965	0.095323	0.009683	N/A
10MB	0.087877	2.071248	2.359491	0.033125	0.004771
1MB	0.084171	0.314489	0.670574	0.505098	0.015822
100KB	0.00123	0.049941	0.020808	0.091238	0.066854

Delay 10ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.325762	0.748634	1.045077	0.225142	0.041738
10MB	0.307441	1.516324	3.331589	0.496379	0.192338
1MB	0.298979	1.36809	5.248097	0.808706	0.718897
100KB	0.200694	0.244106	0.610207	1.644544	1.182667

Appendix C: SCPS Pure-Rate F2

Sample Session for 500ms Delay

Receiver

scps_ttcp –b 5900000 –G 0 –F 2 –R 80000000 –f m -r -s

Sender

scps_ttcp –b 5900000 –l 1024 –G 0 –F 2 –R 80000000 –f m –t destination <input file

Buffer Size / Rate

5.9Mbyte/80Mbps for 500ms Delay
2.9Mbyte/80Mbps for 250 ms Delay
125Kbyte/80Mbps for 10ms Delay

Average Throughput

Delay 500 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	35.31805	34.282069	31.351041	26.468136	11.307485
10MB	25.487627	24.964481	22.258822	18.784879	8.878893
1MB	6.612573	6.494976	5.802159	4.547215	2.2120034
100KB	0.778175	0.768471	0.7446952	0.64775	0.397551

Delay 250 Throughput (Mbps)

File Size		1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	36.015478	35.704313	32.772764	29.079432	14.6033857
10MB	30.149503	29.539839	26.985	23.897358	11.171287
1MB	11.255701	10.980451	10.4472912	8.356353	3.96261
100KB	1.51121	1.4850798	1.514442	1.213998	0.761786

Delay 10ms Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	37.89168	37.528255	37.50707	35.389888	23.697951
10MB	36.61547	36.199935	36.303307	34.388343	21.168497
1MB	27.829117	27.837042	27.408649	25.891731	17.633351
100KB	7.494632	7.552061	7.53466	7.423482	6.34945

Standard Deviation

Delay 500ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.245447	0.353532	0.405608	0.677033	0.362799
10MB	0.153669	0.797925	1.053571	0.184345	1.076813
1MB	0.009231	0.457148	0.883536	0.544653	0.55643
100KB	0.000893	0.045618	0.087067	0.134533	0.148341

Delay 250ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.334065	0.560921	1.427178	0.498491	0.44784
10MB	0.292859	0.882992	1.090789	1.260595	2.655654
1MB	0.217978	0.800825	1.004534	0.508412	1.70459
100KB	0.026065	0.167003	0.003047	0.232241	0.264365

Delay 10ms, Standard Deviation

	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.356443	0.510682	0.373739	0.431592	1.217071
10MB	0.416184	0.696348	0.43459	0.483522	5.644758
1MB	0.511913	0.844118	0.560992	0.957072	4.923369
100KB	0.194662	0.217925	0.1857	0.235728	1.367689

Appendix D: SCPS Pure-Rate F0

Sample Session

Receiver

scps_ttcp –b 5700000 –G 0 –F 0 –R 80000000 –f m -r -s

Sender

scps_ttcp –b 5900000 –l 1024 –G 0 –F 2 –R 80000000 –f m –t destination <input file

Buffer Size / Rate

5.7Mbyte/80Mbps for 500ms Delay
3.1Mbyte/80Mbps for 250ms Delay
2Mbyte/80Mbps for 10ms Delay

Average Throughput

Delay 500 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	40.688945	38.542618	32.483322	26.490601	11.404668
10MB	27.422989	26.38231	22.640064	18.877941	8.942233
1MB	6.606405	6.216356	5.349742	4.676208	2.4753
100KB	0.777802	0.778262	0.751898	0.61858	0.388734

Delay 250 Throughput (Mbps)

File Size		1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	42.581896	41.076811	35.275485	29.294448	14.890691
10MB	33.648937	32.78304	28.651341	23.497282	12.911593
1MB	11.254229	11.164455	10.436182	8.288974	4.291074
100KB	1.51525	1.515154	1.483031	1.047881	0.682512

Delay 10ms Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	43.697272	43.432346	42.645123	38.762649	30.635674
10MB	42.668447	42.82409	42.058357	37.998141	29.279971
1MB	36.28129	36.314922	35.496446	30.532993	20.562389
100KB	7.358565	7.481312	7.44738	7.431969	5.704731

Standard Deviation

Delay 500ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.260966	0.66439	0.54773	0.844467	0.386269
10MB	0.141693	1.174695	1.504228	1.437063	1.021881
1MB	0.019221	0.731287	0.881363	0.278325	0.566643
100KB	0.000229	0.00373	0.076792	0.123748	0.158536

Delay 250ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.533388	0.54884	0.52951	0.644373	0.444731
10MB	0.670403	1.078449	1.24467	1.63257	2.367246
1MB	0.232223	0.504958	1.153637	0.920129	1.549453
100KB	0.000789	0.001968	0.120736	0.240689	0.295343

Delay 10ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.192073	0.2403	0.36046	0.658638	1.91311
10MB	0.58498	0.27014	0.388225	0.579493	5.373988
1MB	0.430171	0.300344	0.794238	2.297841	8.547738
100KB	0.271428	0.200939	0.251613	0.272882	2.001184

Appendix E: SCPS Vegas Congestion

Sample Session

Receiver

scps_ttcp –b 5650000 –G 2 –F 2 –R 60000000 –f m -r -s

Sender

scps_ttcp –b 5650000 –l 1024 –G 2 –F 2 –R 60000000 –f m -t destination <input file

Buffer Size / Rate

5.65Mbytes/60Mbps for 500ms Delay
2.85Mbyte/80Mbps for 250ms Delay
120Kbyte /80 Mbps for 10ms Delay

Average Throughput

Delay 500 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	30.108677	6.345679	0.876501	0.26425	0.034068
10MB	10.960991	9.91819	3.818672	0.274018	0.074636
1MB	1.574237	1.565099	N/A	N/A	N/A
100KB	0.224753	0.226869	N/A	N/A	

Delay 250 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	33.689549	N/A	1.795871	0.523202	
10MB	18.050995	N/A	4.598804	0.53408	0.142618
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Delay 10ms Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	38.458161	N/A	31.1326113	10.5089	1.291429
10MB	36.899317	N/A	31.163916	10.571247	1.316221
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Standard Deviation

Delay 500ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.28778	3.3689	0.059152	0.007405	0.003039
10MB	0.117258	1.909655	3.426908	0.020805	0.002465
1MB	0.000461	0.034786	N/A	N/A	N/A
100KB	0.012394	0.00996	N/A	N/A	N/A

Delay 250ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.236773	N/A	0.163221	0.14193	
10MB	0.052543	N/A	4.170496	0.41496	0.005551
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Delay 10ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.51826	N/A	1.159909	0.209781	0.049272
10MB	0.52149	N/A	2.680634	0.765191	0.179452
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Appendix F: SCPS Vegas Corruption

Sample Session
  Receiver
    scps_ttcp –b 6000000 –G 2 –F 2 –R 60000000 –f m -r -s
  Sender
scps_ttcp –b 6000000 –l 1024 –G 2 –F 2 –R 60000000 –f m -t destination <input file

Buffer Size / Rate

6.0Mbyte/60Mbps for 500ms Delay
2.9Mbyte/80Mbps for 250ms Delay
125Kbyte/80Mbps for 10ms Delay

Average Throughput

Delay 500 Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	30.128228	N/A	12.024788	1.379327	0.22999
10MB	11.797008	N/A	8.430845	2.718787	0.260366
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Delay 250
throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	33.129125	N/A	21.844436	2.145097	
10MB	19.033725	N/A	14.784382	4.214825	0.482859
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Delay 10ms Throughput (Mbps)

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	36.892847	N/A	36.551005	34.387077	9.387179
10MB	36.110538	N/A	35.792866	33.684291	9.760756
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Standard Deviation

Delay 500ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.276621	N/A	4.980335	0.114026	0.026506
10MB	0.012234	N/A	1.997051	1.128122	0.074822
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Delay 250ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.335879	N/A	6.591755	0.146559	
10MB	0.058977	N/A	3.295621	1.737706	0.030398
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Delay 10ms, Standard Deviation

File Size	0	1.00E-08	1.00E-07	1.00E-06	1.00E-05
100MB	0.344125	N/A	0.40424	2.072751	0.978263
10MB	0.234231	N/A	0.273308	0.517483	1.008569
1MB	N/A	N/A	N/A	N/A	N/A
100KB	N/A	N/A	N/A	N/A	N/A

Appendix G: Theoretical Throughput of Congestion-Based Protocol in Mbps

BER	500ms delay	250ms delay	10ms delay
0	90.85397412	86.31127542	94.63955638
1.00E-08	1.563036126	3.126072251	78.15180629
1.00E-07	0.494370078	0.988740156	24.71850389
1.00E-06	0.156632973	0.313265946	7.83164865
1.00E-05	0.050481816	0.100963632	2.524090811
1.00E-04	0.019048219	0.038096439	0.952410965

Appendix H: Theoretical Throughput of Rate-Based Protocol in Mbps at 500ms delay:

BER	100Mbyte	10Mbyte	1Mbyte	100Kbyte
0	72.10632867	50.47443007	12.61860752	1.484542061
1.00E-08	72.10048397	50.4715661	12.61842851	1.484539583
1.00E-07	72.04794665	50.44581593	12.61681837	1.484517294
1.00E-06	71.52892811	50.19082248	12.60080703	1.268641624
1.00E-05	66.90681992	47.87033729	12.44930051	1.482170637
1.00E-04	46.64586703	36.52070187	11.51837988	1.468044811

At 250ms delay:

BER	100Mbyte	10Mbyte	1Mbyte	100Kbyte
0	73.86501961	60.56931608	21.6318986	2.91198635
1.00E-08	73.85888634	60.56519201	21.63137255	2.911976817
1.00E-07	73.80375617	60.52811639	21.62664124	2.91189106
1.00E-06	73.259228	60.16137959	21.57963962	2.911037365
1.00E-05	68.41836251	56.85772045	21.1390665	2.902875973
1.00E-04	47.37556801	41.52870812	18.58813947	2.849182068

At 10ms delay:

BER	100Mbyte	10Mbyte	1Mbyte	100Kbyte
0	75.63600909	74.96202485	68.82876827	37.85582255
1.00E-08	75.62957821	74.95570806	68.82344282	37.85421154
1.00E-07	75.571774	74.89892897	68.77557113	37.83972482
1.00E-06	75.00094517	74.3381818	68.30247413	37.69606888
1.00E-05	69.93512147	69.35852101	64.07562006	36.37188036
1.00E-04	48.09788692	47.82445084	45.25188503	29.42411075

Appendix I: Testbed System Information

Sending Machines

Sun Ultra II, 200 Mhz, 512 Mb RAM, Solaris 7*
Sun Ultra I, 143 Mhz, 64 Mb RAM, Solaris 7
Sun Ultra 10, 440 Mhz, 132 Mb RAM, Solaris 8

Receiving Machines

Sun Ultra II, 200 Mhz, 256 Mb RAM, Solaris 7*
Sun Ultra 5, 270 Mhz, 256 Mb RAM, Solaris 7
Sun Ultra 10, 440 Mhz, 132 Mb RAM, Solaris 8

Monitor/Tracing Machines (Sender)
Pentium III, 550 Mhz, 64 Mb RAM, NetBSD 1.5*
Pentium II, 450 Mhz, 400 Mb RAM, NetBSD 1.3.2
Pentium Pro, 200 Mhz, 64 Mb RAM, NetBSD 1.5

Monitor/Tracing Machines (Receiver)
Pentium III, 550 Mhz, 64 Mb RAM, NetBSD 1.5*
Pentium II, 450 Mhz, 400 Mb RAM, NetBSD 1.3.2
Mac PPC 604, 120 Mhz, 16 Mb RAM, NetBSD 1.5

Additional Networking Equipment
Adtech SX/14 Channel Simulator
CISCO Catalyst 2900 Switch (two)
CISCO 7100 Router (two)

Memory on Terrestrial side router

	head	Total (Byte)	Used (Byte)	Free (Byte)	Lowest (Byte)	Largest (Bytes)
Processor	61F6A440	25779136	9330364	16448772	16332616	16348656
I/O	20000000	67108872	3673752	63435120	63435120	63435068
I/O-2	3800000	8388616	2716696	5671920	5671920	5671868

Memory on Space side router

	head	Total (Byte)	Used (Byte)	Free (Byte)	Lowest (Byte)	Largest (Bytes)
Processor	61F6A440	25779136	9331260	16447876	16331880	16348464
I/O	20000000	67108872	3673752	63435120	63435120	63435068
I/O-2	3800000	8388616	2716696	5671920	5671920	5671868

Buffer pools on terrestrial side routers

```
. Buffer elements:
  499 in free list (500 max allowed)
. Public Buffer pools:
 Small buffer, 104 bytes (total 256, permanent 256)
  250 in free list (64 min, 1280 max allowed)
 Middle buffer, 600 bytes (total 256, permanent 256)
  254 in free list (64 min, 1280 max allowed)
 Big buffer, 1524 bytes (total 256, permanent 256)
  256 in free list (64 min , 1280 max allowed)
 Very big buffer, 4520 bytes (total 256, permanent 256)
  256 in free list (64 min, 1280 max allowed)
 Large buffer, 5024 bytes (total 256, permanent 256)
  256 in free list (64 min, 1280 max allowed)
 Huge buffer, 18024 bytes (total 16, permanent 16)
  16 in free list (4 min, 64 max allowed)
. Interface buffer pools:
  IPC buffers, 4096 Bytes
. Header pools:
    Header buffer, 0 bytes (total 511, permanent 256)
      255 in free list (256 min, 1024 max allowed)
. Particle clones:
    1024 clones
. Public particle pools:
    F/S buffers, 128 bytes ( total 512, permanent 512)
    0 in free list (0 min, 512 max allowed)
    Normal buffers, 512 bytes (total 1024, permanent 1024)
    1024 in free list (512 min, 2048 max allowed)
. Private particle pools:
    FastEthernet 0/0 buffer, 512 bytes (total 400, permanent 400)
     0 in free list (0 min, 400 max allowed)
    FastEthernet 0/1 buffer, 512 bytes ( total 400, permanent 400)
     0 in free list ( 0 min, 400 max allowed)
    ATM 1/0 buffer, 512 bytes ( total 1200, permanent 1200)
     0 in free list ( 0 min , 1200 max allowed)
    ATM 2/0 buffer, 512 bytes ( total 1200, permanent 1200)
     0 in free list ( 0 min, 1200 max allowed)
```

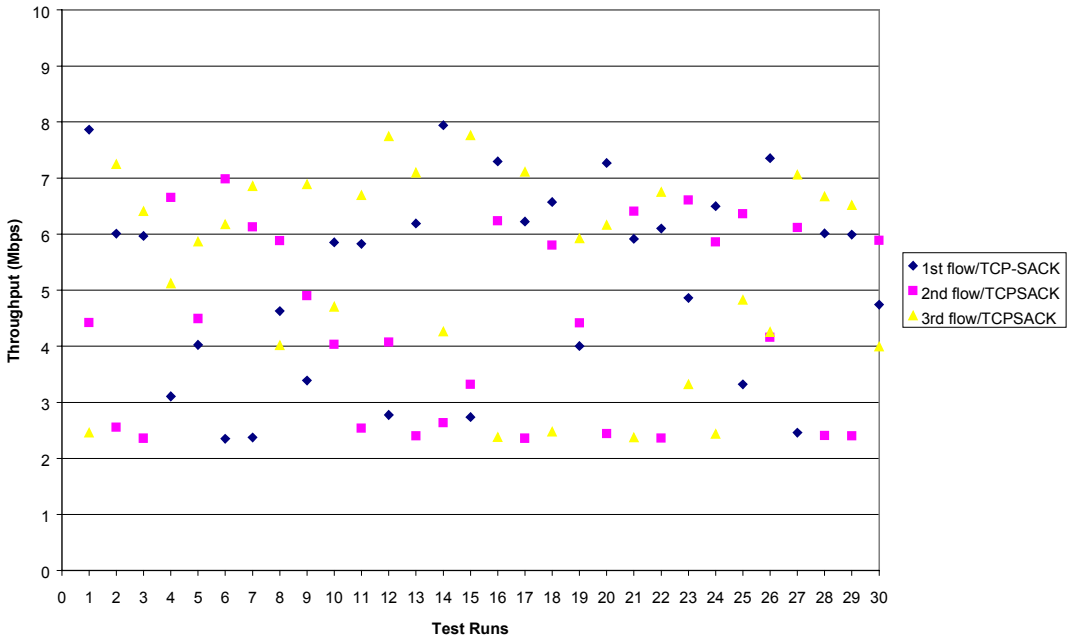
Buffer pools on space side router

```
. Buffer elements:
  499 in free list (500 max allowed)
. Public Buffer pools:
 Small buffer, 104 bytes (total 256, permanent 256)
  252 in free list (64 min, 1280 max allowed)
```

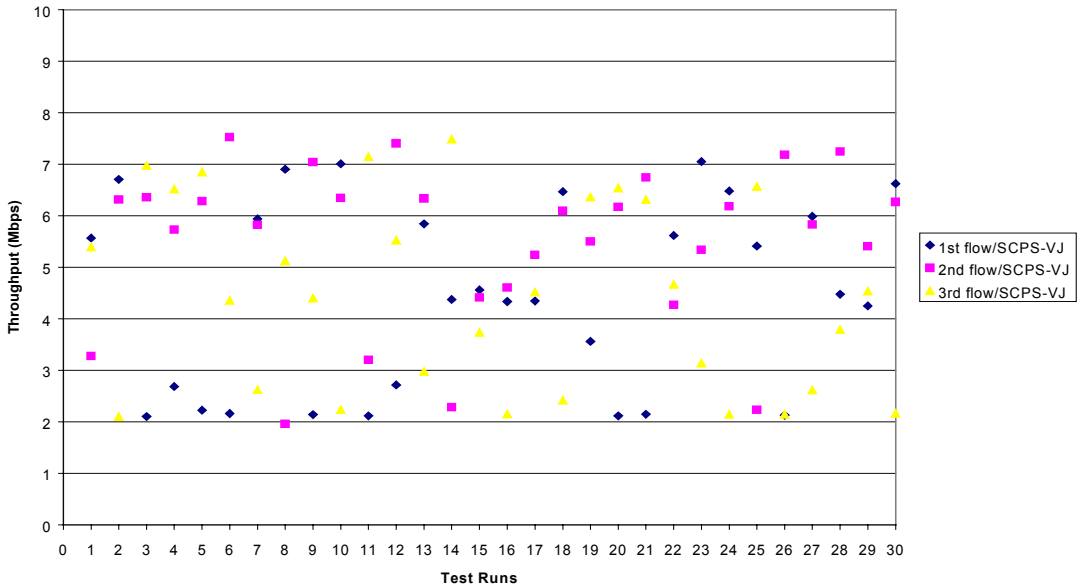
```
Middle buffer, 600 bytes (total 256, permanent 256)
 252 in free list (64 min, 1280 max allowed)
Big buffer, 1524 bytes (total 256, permanent 256)
 256 in free list (64 min , 1280 max allowed)
Very big buffer, 4520 bytes (total 256, permanent 256)
 256 in free list (64 min, 1280 max allowed)
Large buffer, 5024 bytes (total 256, permanent 256)
 256 in free list (64 min, 1280 max allowed)
Huge buffer, 18024 bytes (total 16, permanent 16)
 16 in free list (4 min, 64 max allowed)
. Interface buffer pools:
 IPC buffers, 4096 Bytes
. Header pools:
  Header buffer, 0 bytes (total 511, permanent 256)
    255 in free list (256 min, 1024 max allowed)
. Particle clones:
  1024 clones
. Public particle pools:
  F/S buffers, 128 bytes ( total 512, permanent 512)
  0 in free list (0 min, 512 max allowed)
  Normal buffers, 512 bytes (total 1024, permanent 1024)
  1024 in free list (512 min, 2048 max allowed)
. Private particle pools:
  FastEthernet 0/0 buffer, 512 bytes (total 400, permanent 400)
   0 in free list (0 min, 400 max allowed)
  FastEthernet 0/1 buffer, 512 bytes ( total 400, permanent 400)
   0 in free list ( 0 min, 400 max allowed)
  ATM 1/0 buffer, 512 bytes ( total 1200, permanent 1200)
   0 in free list ( 0 min , 1200 max allowed)
  ATM 2/0 buffer, 512 bytes ( total 1200, permanent 1200)
   0 in free list ( 0 min, 1200 max allowed)
```

* equipment used in Single Flow Tests. All networking equipment were used in both the Single and Multiple Flow Tests.

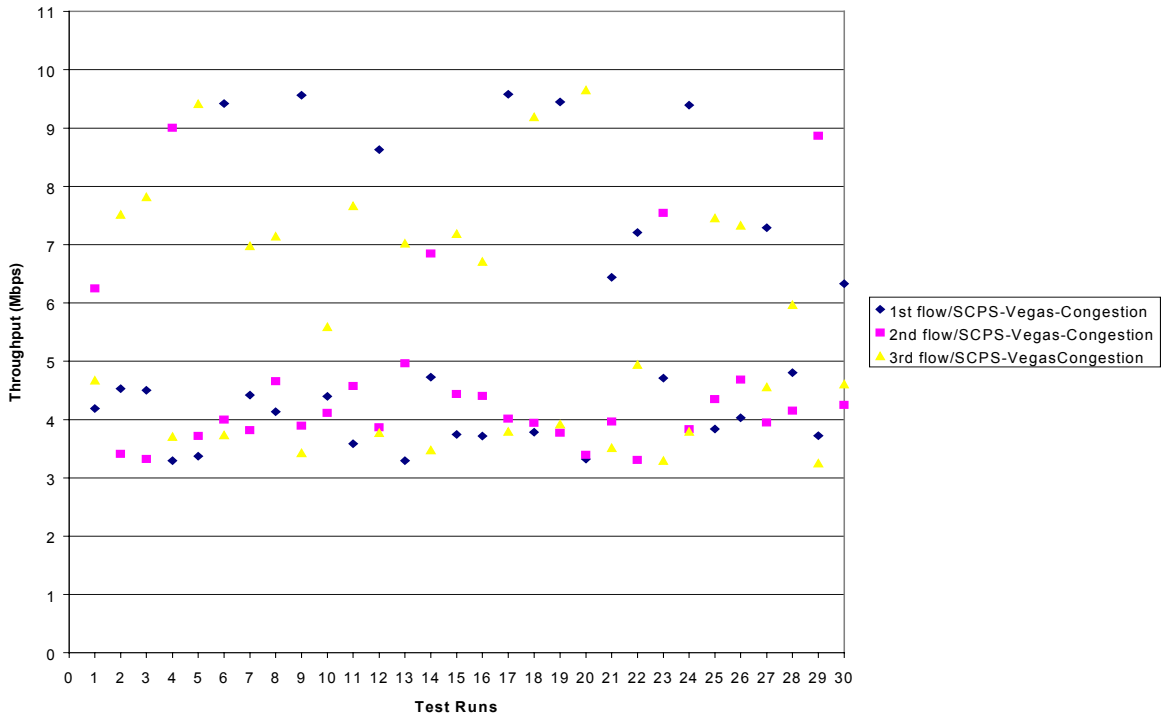
Appendix J: Individual Multiple Flow Transfers



TCP-SACK Throughput vs. Test Runs



TCP-SCPS-VJ Throughput vs. Test Runs



TCP-SCPS-Vegas Throughput vs. Test Runs