MTR 98B0000011

MITRE TECHNICAL REPORT

Space Communications Protocol Standards (SCPS) FY97 DOD Test Report

February 1998

John Muhonen
Robert C. Durst

Sponsor:	USSPACECOM/J4P	**Contract No.:**	F19628-94-C-0001
Dept. No.:	D610	**Project No.:**	6380

MITRE

Center for Air Force C3 Systems

Abstract

The DOD has joined into a cooperative effort with the National Aeronautics and Space Administration (NASA) and the National Security Agency (NSA) to develop the Space Communication Protocol Standards (SCPS). This set of protocols has the potential to increase the efficiency and reliability of data transfer, increase interoperability with both DOD and non-DOD assets, and decrease the cost of operating our space systems. The protocols also have potential applicability to military-tactical and mobile communication environments

Two experiments and a number of simulations were conducted in FY96 to evaluate the performance of a subset of the SCPS protocols. This report summarizes the results of a third test program which was developed in FY97 to further evaluate the performance of SCPS. Although this test program did address functional testing of FP and NP, the focus was on the end-to-end performance of SCPS TP and commercial TCP in networks that include at least one satellite communications link.

In the presence of corruption on the satellite link, we concluded that SCPS TP performs significantly better than TCP for high bandwidth-delay product links. The performance gain was most pronounced for a high bit error rate (BER) and small packet sizes, but it was still significant for very low BER and large packets. For smaller bandwidth-delay product links, the performance of SCPS TP is still better than TCP, although the gain is not as large. In the network congestion environment, the performance of each was similar.

KEYWORDS: SCPS, TCP, IP, FTP, Satellite Communications, end-to-end performance

Executive Summary

Purpose of Report

The purpose of this report is to clearly state the near-term applicability of the Space Communications Protocol Standards (SCPS) to military satellite communications (SATCOM) systems and users based on a test program that was conducted in FY97. To support this goal, this test report summarizes the results of the FY97 test program and, in conjunction with test plans and procedures, documents the activities which led to the collection of the information.

In general, the main sections of the report are primarily written for those potential users and system program offices that are trying to understand how SCPS may benefit their programs in the near term (0-5 years). Appendix A, however, presents a detailed analysis of selected results from the transport protocol testing. This appendix, written more for the protocol developer, describes the inner workings of these protocols and discusses why we obtained the results we did.

Strategic Test Program Objectives

The primary objective of this muli-year test program is to demonstrate the potential utility of SCPS to the DOD user community for two general satellite applications: Tracking, Telemetry, and Command (TT&C) and satellite communications (SATCOM). For the purposes of this test program, TT&C is defined as any application for which the satellite is either the source or destination of the data. Satellite control (both payload and platform) data is clearly TT&C, but mission data (warning, navigation, and environmental) is also viewed as TT&C within the context of this definition. SATCOM, on the other hand, is defined as any application for which the ground is both the source and destination of the data. It is the intent of this test program to not only demonstrate the functionality of SCPS in these two applications but also to quantify the performance of SCPS and identify user resources required in order to use these protocols. The near-term (FY97) test activities focus on the SATCOM application, as discussed next.

FY97 Test Program Objectives

FY97 testing supported the strategic objective of demonstrating the potential utility of SCPS to the DOD user community for the SATCOM application. The functionality and performance of SCPS in this application were demonstrated in a scenario that will potentially benefit existing DOD SATCOM users.

The classical military SATCOM scenarios are: 1) a single user with a satellite terminal wants to communicate with another single user (point-point), and 2) a single user wants to communicate simultaneously with many users (broadcast). Error correction is typically implemented at the physical layer on a link-by-link basis. In these scenarios, no network protocols are required due to the simplicity of the networks, and no end-to-end reliability (transport layer) is implemented. However, there is a growing base of DOD information exchanges utilizing networks on the ground which typically use the TCP/IP set of protocols. Some of these exchanges may involve the use of a satellite link to provide connectivity to geographically separated portions of the network. Using TCP over the typical satellite link may result in poor throughput due to various combinations of: high data rates, relatively high error rates, and large propagation delays.

SCPS has the potential to improve performance in this environment, and users/network managers can determine which of the SCPS protocols are most beneficial for each scenario. The primary goals of this test program were to: 1) quantify the performance of SCPS TP relative to TCP in various environments, and 2) demonstrate a portion of the functionality of FP. A Secondary goal was to demonstrate selected NP and SP functionality as time and resources permitted. The rationale for these goals is described in the report.

Summary of Test Results

Transport Protocol

The performance of SCPS TP relative to TCP was evaluated separately in this test effort for link corruption and network congestion environments.

In the corruption environment with congestion control turned off, SCPS-TP always outperforms TCP over large bandwidth-delay product links (tested here on a 2 Mbps transponder in geosynchronous orbit). Even with no bit errors on the link, TP performs better than TCP due mostly to the slow-start congestion algorithm used with TCP. This performance improvement is significant even for relatively large packets, but it becomes substantial for smaller packets. And, the performance gains summarized herein would be much greater if TP is compared to a version of TCP that does not implement the window scaling option. As the BER on the link increases, the performance improvement of TP relative to TCP in this environment also increases (even for large packets) due mostly to TP's ability to respond to bit errors as corruption, not congestion. Even when the congestion control algorithm for TP is activated, TP still outperforms TCP in all cases except for when large packets are being transmitted and there are no bit errors on the link. For these conditions, TCP performs only marginally better than TP due to the differences in their congestion control algorithms.

When the data rate on the link is reduced substantially (9600 bps in our case), the performance gains of TP relative to TCP are not substantial. For larger file sizes and packets, the performance is nearly identical at a low BER. As the link BER increases, TP has a reasonable advantage over TCP. However, for the typical modulation and coding used on military SATCOM links, this performance gain can be neutralized by a small increase (less than 0.5 dB) in signal-to-noise ratio on the physical link. The relative performance of TP compared to TCP at this data rate is similar for small files and packets, except that TP has a slight advantage over TCP even at a low BER.

In the congestion environment, the current implementation of SCPS TP performs similar to TCP at the high data rate regardless of file or packet size. TCP may have a slight advantage at very low congestion levels due to the differences in the congestion control algorithms, but this seems to get reversed at higher levels of congestion.

At the lower data rate, TP and TCP appear to perform nearly identical for larger packet sizes. When a smaller packet size is used, TP appears to have a slight advantage over TCP at all levels of congestion.

File Handling Protocol

All record update and file transfer operations were successfully demonstrated, although manual interrupt/restart and automatic interrupt/restart functions were never completed. A problem was discovered with the implementation of the "sockets" programming interface in the course of this testing, and we did not have sufficient resources on this test program to resolve the problem.

Network Protocol

The function of NP signaling to TP in response to corruption or congestion was successfully demonstrated. The function of NP packet precedence was also successfully demonstrated by showing that the lowest priority packets always resulted in the longest delays, while the highest priority packets always resulted in the shortest end-to-end delays.

Security Protocol

The intent of this test effort was to functionally demonstrate a subset of SCPS SP features. However, sufficient resources did not exist to enable us to conduct these tests.

Conclusions

This test program was implemented in order to show the utility of SCPS to existing and near-term DOD SATCOM applications. One of the biggest potential benefits of SCPS in these near-term scenarios is provided by TP. The use of this protocol can result in reduced end-to-end delays and increased throughput on corrupted links (SATCOM, in general) with large bandwidth-delay products. When the link data rate or the propagation delay is small, the performance gains are marginal. This has implications for DOD TT&C links as well because they tend to be lower data rate links. However, we can expect data rates for TT&C links to increase in the future as the demand for more capacity and more services increases. From this and previous test efforts, we know that significant improvements in throughput are obtainable using SCPS-TP on transponded geosynchronous links starting at data rates somewhere between 10 kbps and 200 kbps. Given the limited data we have, it is difficult to say at what point the performance gain is significant, and this will depend on the requirements of each user. Further testing and simulation should be conducted in conjunction with a more comprehensive assessment of potential user requirements.

SCPS FP can benefit those near-term SATCOM users who need to transfer files or individual records of files. As demonstrated in this test program, the ability to update records instead of entire files can be of great benefit in resource-constrained environments (for example, a low data rate link with a short access time). Although not successfully demonstrated in this test program, the ability to automatically restart a file transfer after it is interrupted (due to a link outage or other interruption in service) can be a significant benefit to all DOD data transfer applications. Within the constraints of each potential application, future users should consider implementing SCPS within the kernel of the operating system to avoid some of the difficulties encountered in this test program.

SCPS NP can benefit some near-term SATCOM users (depending on the scenario) by providing the capability to signal the presence of corruption or congestion to the transport layer. Probably the most significant benefit for the near-term SATCOM users is the ability to enforce packet precedence, which allows higher priority traffic to get through a congested network.

Near-term SATCOM users can also benefit from the end-to-end security services provided by SCPS SP. None of these services were demonstrated in this test program due to limited project resources.

SMC, the DOD, NASA, and the International Organization for Standardization (ISO) should continue to seek out near-term applications for SCPS as well as far-term ones. Although the FY97 SCPS DOD test program focused on near-term SATCOM applications, we should not lose sight of potential future applications of SCPS to DOD operations. SMC is currently engaged in a study, with support from MITRE, to identify programs and classes of programs that can be expected to benefit the most (technically) from implementing some or all of the SCPS protocols.

There is also a benefit, although sometimes less tangible, of standardization. Standardization has the potential for cost savings due of commonality among systems, but this potential is not always realized. In addition, standardization can promote interoperability, which more often increases capability but can also reduce cost. The four SCPS protocols are currently in process as formal military standards and as ISO standards. In addition, SCPS has been added to the current version of the Joint Technical Architecture (JTA).

More information on SCPS can be obtain at the following web site: http://www.scps.org/scps. All of the SCPS documentation is available at this site, as well as points of contact and instructions on how to obtain a copy of the reference implementation that was used in this test program.

Acknowledgments

The authors would like to acknowledge Pat Feighery of the MITRE Corporation for his commitment and dedication in support of this test program. In particular, Mr. Feighery conducted all of the testing for the network and file protocols and much of the testing for the transport protocol. Additionally, Pat played an instrumental role in identifying and correcting problems with the protocols and the test tools. We would also like to acknowledge everyone at Rome Laboratories for making this test program possible. First, we would like to thank Steven Yax and management of the Communications Directorate for supporting this effort at no cost to the SMC program office. Second, we would like to thank Mr. Craig Evens, Mr. David Legare, Mr. Peter Radesi, and Mr. Richard Smith for their professionalism and individual dedication to this test program. A special thanks goes to Mr. Evens and Mr. Smith for supporting many hours of late-night tested, which was crucial to obtaining performance results on one of the protocols.

Finally, we would like to acknowledge Dave Collette and Elizabeth Newmiller, both of the MITRE Corporation, for assisting in summarizing performance data for presentation and final editing/production of the report, respectively.

Table of Contents

Section **Page**

List of Figures

Figure **Page**

List of Tables

Section 1

Introduction

1.1 Purpose of Report

The purpose of this report is to clearly state the near-term applicability of the Space Communications Protocol Standards (SCPS) to military satellite communications (SATCOM) systems and users based on a test program that was conducted in FY97. To support this goal, this test report summarizes the results of the FY97 test program and, in conjunction with test plans and procedures, documents the activities which led to the collection of the information.

In general, the main sections of the report are primarily written for those potential users and system program offices that are trying to understand how SCPS may benefit their programs in the near term (0-5 years). Appendix A, however, presents a detailed analysis of selected results from the transport protocol testing. This appendix, written more for the protocol developer, describes the inner workings of these protocols and discusses why we obtained the results we did.

1.2 Background

The DOD has joined into a cooperative effort with the National Aeronautics and Space Administration (NASA) and the National Security Agency (NSA) to develop SCPS. The DOD portion of this effort was originally managed by USSPACECOM/J4P; however, this responsibility was transferred to the Space and Missiles System Center (SMC) during the latter part of FY96. From the DOD viewpoint, this protocol set will increase the efficiency and reliability of data transfer, increase interoperability with both DOD and non-DOD assets, and decrease the cost of operating our space systems.

SCPS consist of a set of four protocols that operate at the network layer and above of the Open Systems Interconnect (OSI) model. The File Handling Protocol (FP) is an application layer protocol (layer 7 in the OSI model) that was derived from the Internet file transfer protocol (FTP). FP is more capable than FTP in that individual records within a file can be updated in addition to the entire file. Another important feature of NP is that a file transfer can be automatically restarted after an interruption. The Transport Protocol (TP) is a transport layer protocol (layer 4 in the OSI model) that was derived from the Internet transmission control protocol (TCP). TP incorporates many of the recent improvements to TCP (window scaling, timestamps, and the experimental congestion control mechanism known as TCP Vegas) as well as some SCPS-defined enhancements, such as selective negative acknowledgment and rate control. This allows TP to provide better end-end throughput in the space environment by providing different responses to congestion and corruption. The Security Protocol (SP) is based on the security protocol at layer 3 (SP3) and the network layer security protocol (NLSP) with reduced overhead. SP does not have a corresponding layer in the OSI sense, rather it operates between the network and transport layers (layers 3 and 4). The Network Protocol (NP), as the name implies, is a network layer protocol (layer 3 in the OSI model) that was developed to be a bit-efficient, scaleable protocol for a broad range of spacecraft environments. Among other things, NP provides for a selectable routing method, connectionless and managed connection operations, corruption and congestion signaling to TP, and handling of packet precedence.

Two experiments and a number of simulations were conducted in FY96 to evaluate the performance of a subset of the SCPS protocols. The SCPS Bent-Pipe Experiment implemented a Space-Ground Link System (SGLS) transponder on a M-22 payload to relay data between two ground nodes located in Sunnyvale, California. During this experiment, the performance of TP was tested under various conditions of BER (Bit Error Rate), packet size, and other TP parameters. The SCPS Space Technology Research Vehicle (STRV) Experiment tested a portion of SCPS over a link between Lashum, England and the United Kingdom Defense Research Agency's STRV-1b spacecraft. Limited functional tests were conducted for FP and SP, while both functional and performance testing of TP was conducted. A physical problem with the antenna on this spacecraft severely limited the ability to perform tests on this vehicle.

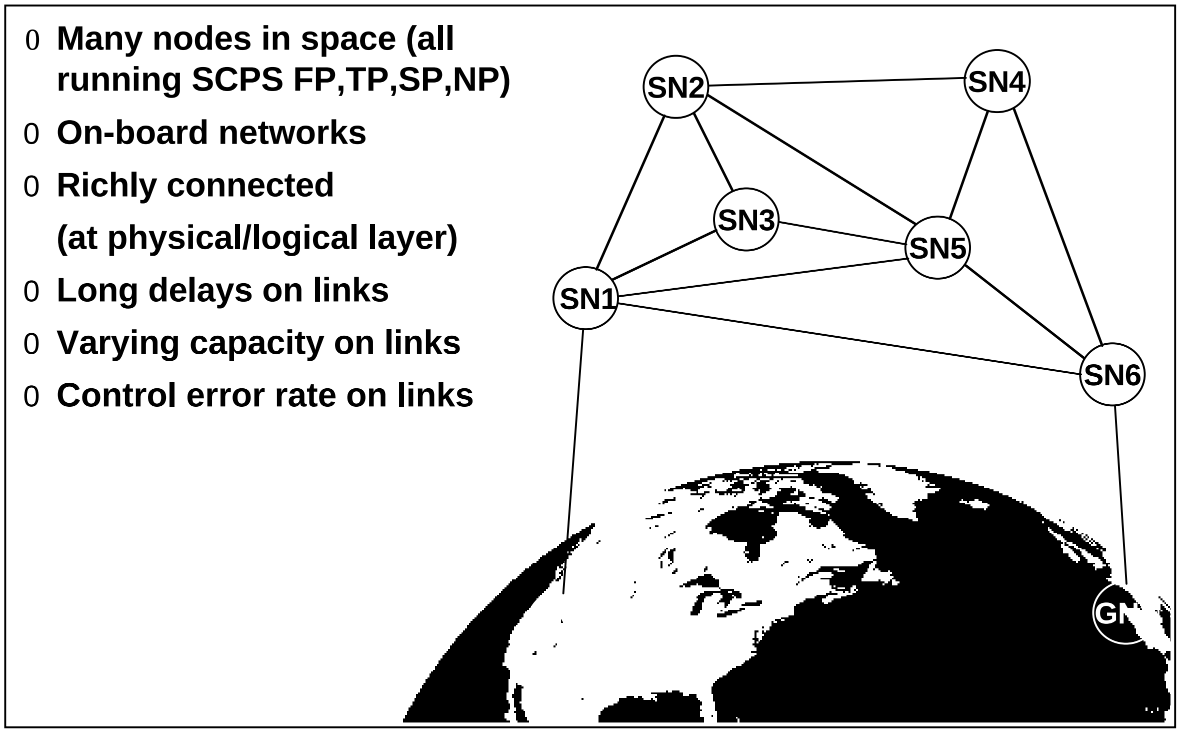
1.3 Strategic Test Program Objectives

The primary objective of this muli-year test program is to demonstrate the potential utility of SCPS to the DOD user community for two general satellite applications: Tracking, Telemetry, and Command (TT&C) and satellite communications (SATCOM). For the purposes of this test program, TT&C is defined as any application for which the satellite is either the source or destination of the data. Satellite control (both payload and platform) data is clearly TT&C, but mission data (warning, navigation, and environmental) is also viewed as TT&C within the context of this definition. SATCOM, on the other hand, is defined as any application for which the ground is both the source and destination of the data. It is the intent of this test program to not only demonstrate the functionality of SCPS in these two applications but also to quantify the performance of SCPS and identify user resources required in order to use these protocols. The near-term (FY97) test activities focused on the SATCOM application, as discussed in section 2.1.

1.4 Ideal Test Network Topology

The ideal network topology for testing the SCPS protocols is depicted in Figure 1. This is ideal because it would demonstrate the widest range of SCPS capabilities. Here, there are many network nodes in space, all running the full SCPS stack. These nodes are richly connected at the physical or data link layers, and each node may also have extensive on-board networks with individually addressable entities. For the TT&C test scenario, a connection would have one end on the ground while the other end would be at one of the nodes in space. For the SATCOM test scenario, both ends of any connection would be on the ground.

The space segment of this ideal topology does not exist today. Therefore, if we are to demonstrate the utility of SCPS to potential DOD users any time in the near future, we must not only emulate realistic DOD network topologies (existing and near future [5-10 years]), but we must also chose a topology that exists or can be readily constructed.



0 **Many nodes in space (all running SCPS FP,TP,SP,NP)**

0 **On-board networks**

0 **Richly connected (at physical/logical layer)**

0 **Long delays on links**

0 **Varying capacity on links**

0 **Control error rate on links**

Figure 1. Ideal Test Network Topology

1.5 Practical Test Network Topology

A more practical test network topology is depicted in Figure 2. This network can be constructed today, it more closely resembles networks that will exist in the DOD for the next 5-10 years, and it supports both the TT&C application and the SATCOM application.

For TT&C, one end of the network would be on the ground, possibly at Ground Node #1 (GN1), and the other end would be at Space Node #1 (SN1). Due to schedule and budget constraints, this type of test was not possible in fiscal year 1997 (FY97). However, this test would support the overall strategic goals of the test program, and TT&C tests of this nature should be performed in the future as our DOD TT&C networks evolve.

For the SATCOM application, both ends of a network connection would be on the ground, possibly at GN1 and GN7 in Figure 2. For existing and near-future systems, there will not be a network node in space, and SN1 in this figure can be replaced by a physical layer repeater (i.e., a transponder or bent-pipe). Even Milstar, which does all of its processing on user data circuits at the physical and data link layers, can be treated the same (from the network layer and above perspective) as other SATCOM systems once a call is set up. FY97 DOD testing of SCPS directly supported the SATCOM application.

- **Multiple nodes on ground**
  - **All running SCPS**
  - **Connection oriented**
  - **Data rate, error rate, and delay controlled**
- **One node at GEO**
  - **SN1 running SCPS**
  - **Long delay**
  - **Error rate controlled**
- **All comm to SN1 encrypted**
- **Ends defined by TT&C or SATCOM application**

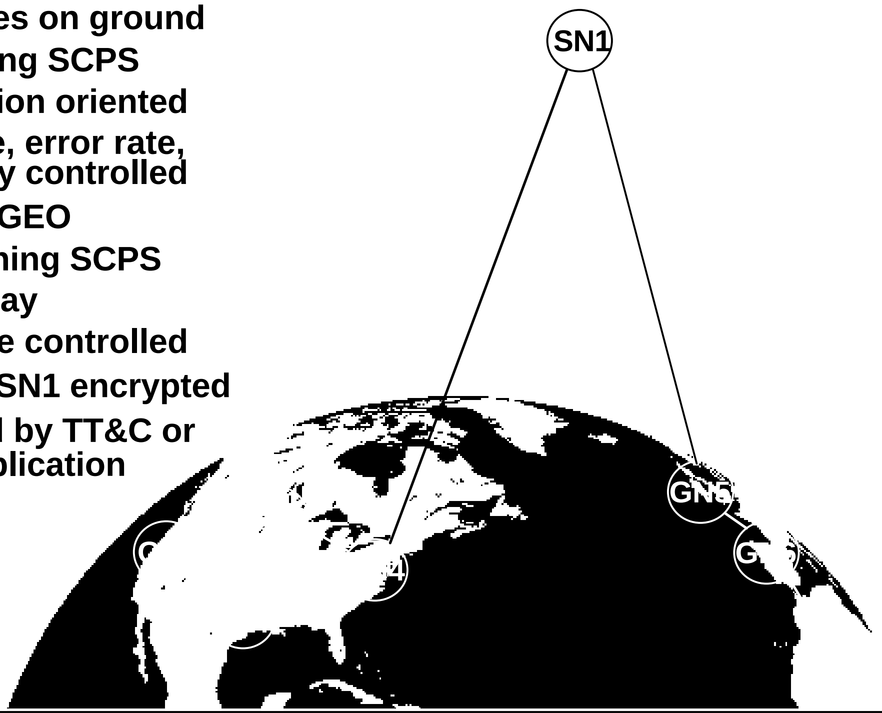


Figure 2. Practical Test Network Topology

1.6 Document Organization

Section 2 defines the FY97 SCPS DOD test program in general terms. Test objectives are stated, test management is defined, and general test resources are identified. Section 3 defines specific tests that were conducted. For each protocol test, the protocol requirements to be verified are presented, the overall experiment design is described, data reduction is discussed, and a summary of the results is presented. Section 4 presents an overall test summary, and Section 5 states the conclusions from this test effort.

Section 2

FY97 Test Program Definition

2.1 Specific FY97 Test Objectives

FY97 testing supported the strategic objective of demonstrating the potential utility of SCPS to the DOD user community for the SATCOM application. The functionality and performance of SCPS in this application were demonstrated in a scenario that will potentially benefit existing DOD SATCOM users.

The classical military SATCOM scenarios are: 1) a single user with a satellite terminal wants to communicate with another single user (point-point), and 2) a single user wants to communicate simultaneously with many users (broadcast). Error correction is typically implemented at the physical layer on a link-by-link basis. In these scenarios, no network protocols are required due to the simplicity of the networks, and no end-to-end reliability (transport layer) is implemented. However, there is a growing base of DOD information exchanges utilizing networks on the ground which typically use the TCP/IP set of protocols. Some of these exchanges may involve the use of a satellite link to provide connectivity to geographically separated portions of the network. Using TCP over the typical satellite link may result in poor throughput due to various combinations of: high data rates, relatively high error rates, and large propagation delays.

SCPS has the potential to improve performance in this environment, and users/network managers can determine which of the SCPS protocols are most beneficial for each scenario. For example, SCPS TP generally exhibits better throughput than typical commercial implementations of TCP in the space environment for the following reasons. First, TP allows for window scaling so that a larger number of bytes can be in transit (end-to-end) before an acknowledgment is sent to release the next segment. This is an option available on TCP that is not always implemented. Second, TP is able to respond to corruption (e.g. SATCOM link) in addition to congestion, so it does not automatically back off the transmission rate and initiate the congestion control algorithms (that TCP does) when errors are due to corruption. Third, TP uses selective negative acknowledgment (SNACK) to identify those packets in error to be retransmitted. TCP, with its cumulative acknowledgment mechanism, does not have the ability to signal missing segments beyond the one being acknowledged. If more than one packet is corrupted, TCP is severely limited in its ability to recover in an environment with long delays (i.e., after the first is retransmitted, a round trip delay will pass before TCP is informed of any subsequent losses). A selective acknowledgment (SACK) option for TCP is forthcoming, which will provide similar information to the SCPS SNACK option but in a less bit-efficient manner. Finally, SCPS TP allows for header compression, which can also reduce the overhead associated with a connection. Compared to the other SCPS protocols, TP has the largest potential for improved performance in this environment, and as such, the minimum goal of FY97 testing was to demonstrate this improvement over commercial TCP.

SCPS FP may be desirable to a SATCOM user if that user has a need to transfer files or individual records of files and may want the protocol to automatically resume the transfer after it was interrupted. Therefore, the goal was to partially demonstrate this FP functionality, but because not all SATCOM users will have these needs, detailed performance testing with comparison to commercial FTP was not planned.

In addition to other potential benefits, SCPS NP provides for packet precedence, signaling to TP for corruption, and less overhead than IP. These functions may or may not be important to any given user in this SATCOM environment. On one extreme, if no packet precedence is needed, if the network is relatively simple, and if only corruption can be expected on the satellite link (no network congestion), then NP is not needed (i.e., TP can be implemented over IP and set to respond only to corruption). On the other extreme, if all the features of NP are desired, they can be made available if there is willingness to install SCPS at each node in the network. In other cases, users may want to realize the potential benefits of SCPS over a satellite link, but they may have to interface with existing TCP/IP-based networks on the ground. Even in this scenario, TP/NP encapsulated by IP can provide better performance than TCP/IP in the corruption environment. In the test planning phases, we tried to evaluate what near-term scenarios would be applicable to most users as we structured individual tests. The resulting goal of FY97 testing was to evaluate selected NP functionality as time and resources permitted.

The SCPS SP protocol provides end-to-end security services. Physically, it resides between the network and transport protocol layers. The end points protected by SP can vary widely. For the TT&C application, an end point could be an instrument operator on the ground connecting to an onboard instrument or a ground control center connecting to a data handling front-end aboard a spacecraft. For the SATCOM application, the end point might be only within the ground network and link security services (e.g., discrete encryption devices) might be used to protect the space-ground link. Because SP is algorithm-neutral (algorithm choices are left as a local security implementation decision), the objective in FY97 was to functional test SP as time and resources permitted.

Table 1 identifies the priority of SCPS FY97 testing applied in support of the SATCOM application. Note that in the second column, a designation of TP/IP means that the higher layer protocol TP is running on top of IP. And, the designation [SCPS]IP indicates that FP/TP/SP is running on top of NP which is encapsulated by IP (a protocol running at the same layer as NP). Specific configurations under which the protocols were tested are defined in section 3.

Table 1. Priority of FY97 Protocol Tests

Protocol Under Test	Protocol Environment	Test Configuration	Type of Test
TCP	TCP/IP	1	P
TP	TP/IP	1	F, P
FP	FP/TP/IP	1	F
NP	[TP/NP]IP	2	F
SP	[SCPS]IP	2	F

F=functional test, P=performance test

2.2 Test Resource Selection

A number of platforms and ground networks were considered in support of the FY97 DOD SATCOM test objectives. The primary platforms under consideration were: the Defense Satellite Communication System (DSCS), Milstar, Fleet Satellite Communication System (FLTSAT), and Advanced Communication Technology Satellite (ACTS). The primary networks were the MITRE, Bedford, Milstar networks laboratory and various Rome Laboratory networks, although others were considered.

After careful consideration, we chose Rome Laboratory as the site to support FY97 DOD SCPS testing over an ACTS link. The primary reasons for this decision were the relative flexibility of the Rome Labs resources (wide range of adjustable data rates and bit error rates, numerous network configurations, and the availability of transportable terminals) and the overall lower cost of conducting tests.

2.3 Test Management

2.3.1 Test Director

A representative from SMC/ADC acted as the test director for all FY97 DOD SATCOM testing. The test director oversaw the conduct of all tests and had the authority to proceed or abort test activity on a daily basis. This responsibility was, in general, delegated to MITRE throughout most of the testing.

2.3.2 Test Participants

A representative from The MITRE Corporation acted as the test conductor for all FY97 SATCOM testing. As such, MITRE was responsible for the overall planning, scheduling, and conduct of these tests. Specifically, MITRE:

- Prepared all test plans and procedures;
- Conducted site survey, installed protocol software, developed protocol specific interfaces to network and terminal resources;
- Coordinated with test participants to ensure availability of all test resources, including test documentation, equipment, facilities, and personnel;
- Conducted all tests in accordance with test plans and procedures;
- Redlined test procedures during tests as required;
- Conducted any in-process data reduction or analysis as required; and
- Prepared the report.

A representative of Rome Laboratory was responsible for coordinating all satellite resources, test equipment, and facilities required to perform FY97 SATCOM testing. The Rome Laboratory personnel:

- Reviewed and commented on all test plans and procedures;
- Scheduled satellite resources with NASA;

- Configured all network and terminal equipment and established the physical SATCOM links to the terminals; and
- Assisted in the general conduct of the tests.

2.4 Test Documentation

The formal documentation associated with this effort consists of: a test plan, a set of test procedures, and this test report. The test plan [1] documented the overall test planning effort. It defined the FY97 SCPS DOD test program by identifying test objectives, management, schedule, required test resources, and specific tests to be conducted. The test procedures [2] ensured that each test was conducted as planned and provided adequate documentation of each test to fully understand the conditions under which results were obtained. This test report defines each test performed, summarizes detailed test data collected during testing, explains the results of each protocol test, and provides an overall summary of the test results.

Section 3

Protocol Test Results

3.1 Transport Protocol

SCPS TP testing took the highest priority in FY97 in support of potential DOD SATCOM users. Tests were conducted to demonstrate both the functionality and the performance of TP. For the performance tests, measurements were taken relative to TCP performance under two main categories for the network environment: congestion or corruption. Test were conducted intentionally under separate controlled environments to be able to more clearly state results upon completion of the tests.

3.1.1 Protocol Requirements

Table 2 identifies those functional and performance requirements of TP that were tested in the SATCOM environment. These were derived from [3].

Table 2. SCPS Transport Protocol Requirements

Ref Para	Requirement	Type of Test
T.1	Full reliability (provided there is end-to-end link availability and sufficient link capacity for retransmissions).	N/A
T.1.1	Shall provide the capability to deliver all data segments to the correct destinations, as addressed at the source.	F
T.1.2	Shall provide the capability to deliver all data segments in the same order as originated at the source, with no duplicate or extraneous data.	F
T.1.3	Shall provide the capability to deliver all data segments for which there are no detected errors.	F
T.1.4	Shall provide the capability to recover from detected data transmission errors.	P
T.7	Operation over a wide range of conditions.	N/A
T.7.3	Shall be able to operate reliably under the delay, bandwidth, and error conditions typical of space-based communication environments.	P
T.9	Response to congestion and corruption.	N/A
T.9.1	Shall provide the capability to differentiate between network congestion and network data corruption, as identified by the network level protocol.	F
T.9.2	Shall provide the capability to counteract the identified network congestion anomalies.	P
T.9.3	Shall provide the capability to compensate for the identified network data corruption anomalies.	P

F=functional test, P=performance test

3.1.2 General Experiment Design

All TP performance testing was conducted under two primary network environments. First, TP was tested under various levels of satellite link corruption without ground network congestion. At each level of corruption, parametric data was collected to determine the overall utility of TP to potential near-term DOD users. Specifically, throughput an delay were characterized as a function of data rate, BER, file size, and packet size. This characterization is the focus of this report. In addition, a number of secondary protocol, link, and user traffic factors were varied to collect engineering information of value to the protocol developers. These parameters and other details are discussed further in Appendix A.

Second, TP was tested under various levels of ground network congestion without satellite link corruption. Data rate, TP packet size, and other secondary factors were varied and the same performance parameters were measured.

In order to state performance results of value to both potential users and developers, end-to-end performance was characterized first for TP, then the tests were repeated while operating a commercial version of TCP.

3.1.3 Test Method

3.1.3.1 Corruption Environment

We adopted the general method of using Expect script language from FY96 SCPS TP testing to the extent possible to automate the test procedures. Scripts developed for FY96 SCPS TP testing were modified to maximize efficiency in automating FY97 tests.

The specific network configuration used for TP performance testing is identified in Figure 3. For corruption environment tests, WS1 hosted TCP/IP, TP/IP, and the test drivers associated with the source end of the connection. WS2 hosted TCP/IP, TP/IP, and test drivers associated with the destination end of the connection. During these tests, WS3, which generates congestion traffic, remained connected to the local area network (LAN), but did not generate any traffic. Throughout the testing, the tests were executed and data was collected both locally at Rome Laboratory and remotely from Reston, VA as indicated in Figure 3.

All workstations, routers, satellite terminals, and interconnections were provided by Rome Laboratories. The modems which operated over the ACTS link were Com Stream CM701 digital modems. Most of the protocol tests were conducted over a quadri-phase shift key (QPSK) modulated link; however, some of the low data rate tests were done using binary phase shift keying (BPSK). In all cases, forward error correction was employed that is most common on DOD SATCOM links (rate 1/2, constraint length 7, convolutional coding). The Com Stream modem employs a fairly versatile BER tester, which enabled us to make direct measurements of BER on the SATCOM link being used for the protocol tests. These BER testers, and the modems in general, were controlled remotely through an RS-232 interface.

A general overview of the steps taken to acquire the data summarized in this test report can be found in the test plan [1], and the specific steps taken are described in detail in the test procedures [2].

Note that the configuration tested in Figure 3 is of most wide-spread interest to near-term DOD users and network managers. As stated earlier, some users will be able to benefit from other SCPS protocols and some will not. However, for any network that has at least one satellite link in it, better overall throughput is possible by replacing TCP at the end systems with SCPS TP. As shown in Figure 3, none of the intermediate nodes are required to run any of the SCPS protocols in order to realize the performance gains.

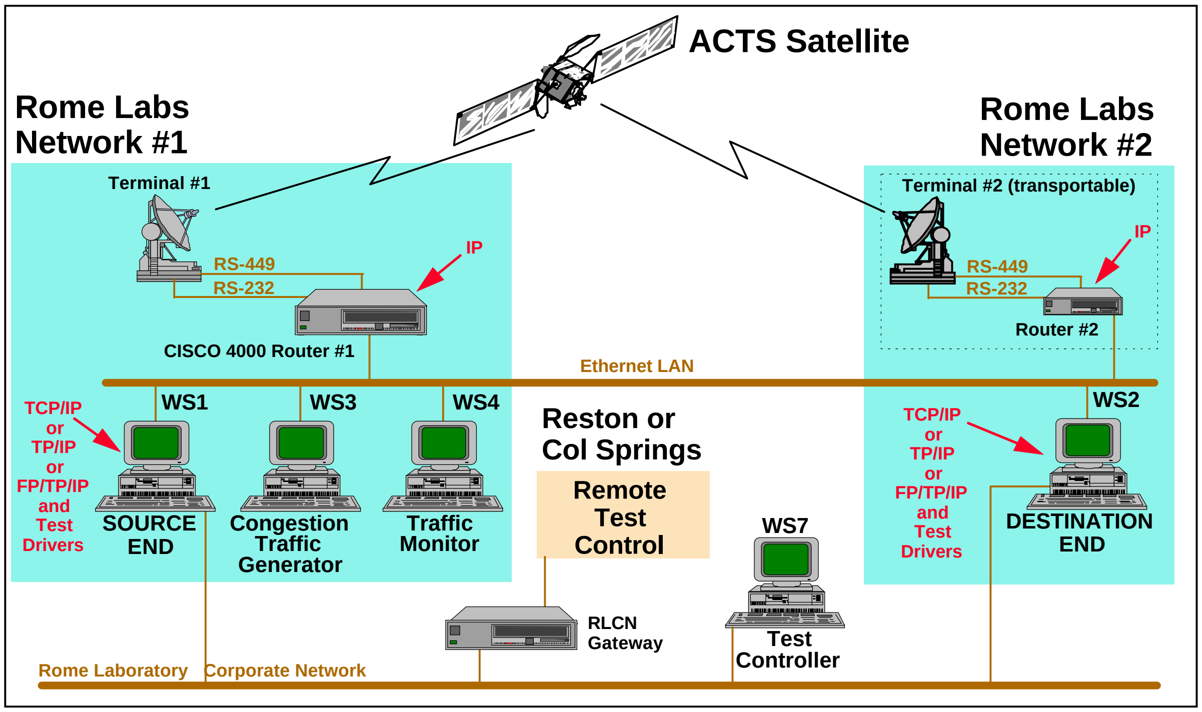


Figure 3. Test Configuration 1

3.1.3.2 Congestion Environment

The specific network configuration identified in Figure 3 was also used for congestion testing of TP. Here, WS1 hosted TCP/IP, TP/IP, and the test drivers associated with the source end of the connection. WS2 hosted TCP/IP, TP/IP, and test drivers associated with the destination end of the connection. WS3 generated random traffic to produce various levels of congestion to the satellite link (and therefore, the RS-449 interface to the CISCO 4000 router). We selected a uniformly distributed (with a minimum of zero and a maximum to be specified in each test) random traffic generator to emulate the aggregate of traffic that might be experienced from a number of independent data sources. This was then combined with the intended user traffic (an adaptive level generated at up to the maximum capacity of the SATCOM link). The resultant traffic presented at the unit being congested, was a uniformly distributed random variable with a minimum equal to the link data rate and a maximum specified in terms of the capacity of the link. For example, if 200% congestion was specified for a test, the maximum aggregate traffic was two times the capacity of the link.

We did not set this test up to be a rigorous assessment of SCPS TP performance in the congestion environment. Rather, we intended this to be a high level verification that SCPS TP would perform on approximately the same level as TCP in this environment.

3.1.4 Data Reduction

For each test, raw data was collected in the form of the following files: *tcpdump* (WS4), the TCP responder log (WS2), and the TP responder log (WS2). Initial data reduction was performed by the test drivers to derive the primary measures of performance (end-to-end throughput and delay). Delay was directly calculated by measuring the elapsed time between when the first packet arrived at the destination and when the acknowledgment of the last packet was sent by the destination. Throughput was then calculated by dividing the amount of data in the transfer by the delay.

Data was collected for multiple test runs under identical conditions; however, each test was repeated only five times to conserve on test resources. Averages and standard deviations were subsequently calculated for each case, and the classical confidence intervals were derived. The Student-t distribution was used instead of the standard normal distribution because the sample size was small and the population is believed to be normal. Therefore, the y-axis error bars displayed for data in the next section represent the 95% confidence bounds using this approach (i.e., the probability is .95 that the actual mean is within the error bounds, given the sample mean and variance).

3.1.5 Summary of Results

SCPS TP and TCP performance results are summarized here separately for link corruption and network congestion environments for a number of reasons, not the least of which is the fact that the results are easier to understand. Also, a number of key issues should be kept in mind when interpreting the results.

First, when we developed the test plan [1] for this effort, we focused our attention on scenarios representative of existing and near-term DOD SATCOM users (less than five years). Through a partial survey of the DOD user community, we found that an increasing number of SATCOM users are connecting into TCP/IP-based networks. In this general scenario, a given user typically understands the SATCOM assets at his disposal, and he does not attempt to congest this resource with more traffic than it can support (although TCP can congest the link without the user knowing it if not configured properly). He also is not in contention with other users for the resource once it is allocated.

Second, although the potential benefits of TP (and SCPS in general) are numerous, the biggest benefit to near-term DOD SATCOM users is increased throughput (reduced delay) on corrupted links (SATCOM, in general) with large bandwidth-delay products. The primary comparison of TP to TCP in the corruption environment is done (in sections 3.1.5.1 and 3.1.5.2) with congestion control turned off for TP. However, comparisons in the corruption environment are also made with TP's congestion control algorithm turned on. Not only does this provide more information in general, this also indicates the performance penalty a user may expect to experience if he anticipates some network congestion and wants to set up TP to be able to respond to it. This will become increasingly important as we: 1) develop future implementations of demand-assigned multiple access (DAMA) in the various military bands, 2) incorporate other means of bandwidth sharing of satellite resources (both connection oriented and connectionless), and 3) establish more communications architectures that consist of combinations of dissimilar networks with various communications media and protocols.

Third, it is important to note that all corruption testing was done under mildly bursty conditions due to atmospherics on the SATCOM link. All data was taken over an ACTS link using BPSK or QPSK modulation and rate 1/2, constraint length 7, convolutional coding. However, the available modem did not implement an interleaver in conjunction with the convolutional coding. This scenario is typical of most DOD SATCOM links except that an interleaver is generally used with such coding. An interleaver will spread out the errors (in time) resulting from a burst error so that they are no longer correlated (if properly designed for the anticipated duration of most bursts). The combination is very effective because convolutional codes are very powerful in response to random errors. Since we did not test with an interleaver, even relatively small variations in signal level caused our error statistics to be more bursty than would be expected on a typical DOD link. In an attempt to mitigate these effects, we performed most of the critical corruption testing during evening and late night hours when the atmosphere was less turbulent. This helped considerably, but it did not completely remove the problem. As a result, both protocols performed somewhat better than would be expected for a channel with uncorrelated, random errors. Due to resource constraints on the project, we were not able to fully characterize the error statistics for this testing, and all error bars on the measured BER in the charts that follow are simply estimates based on the variation in signal level during the testing. Although the channel error statistics were not fully characterized for this testing, a parallel simulation effort was conducted on this project with a random distribution of errors on the channel. This simulation effort will be documented in a separate report.

Fourth, the comparison of TP to TCP in the congestion environment is done in sections 3.1.5.3 and 3.1.5.4. Congestion testing was not as thorough as corruption testing because of the perceived relative importance to near-term DOD SATCOM users.

Finally, it should be noted that the implementation of TCP used for this testing was within the kernel of the operating system, which is typically the case. Operational use of SCPS-TP can also be expected to be implemented this way or via a gateway. Implementation inside the kernel allows for COTS applications (in addition to SCPS-FP) to have SCPS-TP available to them. During the SCPS testing, we experienced some problems in making fair comparisons between an in-kernel TCP and an out-of-kernel TP. On infinitely fast machines, the performance penalty is negligible; however, the workstations used in this test effort were older 486 machines running at 66 MHz. This resulted in a less than optimum implementation for SCPS TP.

For all the above reasons, care should be taken in interpreting how the results presented here relate to specific existing or planned scenarios. In the following sections, results are presented as a family of curves indicating the end-to-end performance of each protocol as a function of the primary independent variable for that environment.

3.1.5.1 Corruption Testing - High Data Rate

For corruption testing, the primary independent variable is the BER on the satellite link. For these tests, throughput and delay were measured as a function of BER for various combinations of two data rates, three packet sizes, and two file sizes.

The high data rate selected for these tests was 2 Mbps [1]. At this data rate, the performance of SCPS TP was compared to that of TCP at three packet sizes. Figure 4 indicates the relative throughput performance of these two protocols when attempting to send a relatively large file (4 Mbyte) over this geosynchronous link using large packets (1400 bytes plus headers) appropriately sized for the file. In this figure, throughput is expressed as a percentage of the maximum link capacity available. Therefore, 100% throughput is equivalent to 2 Mbps.

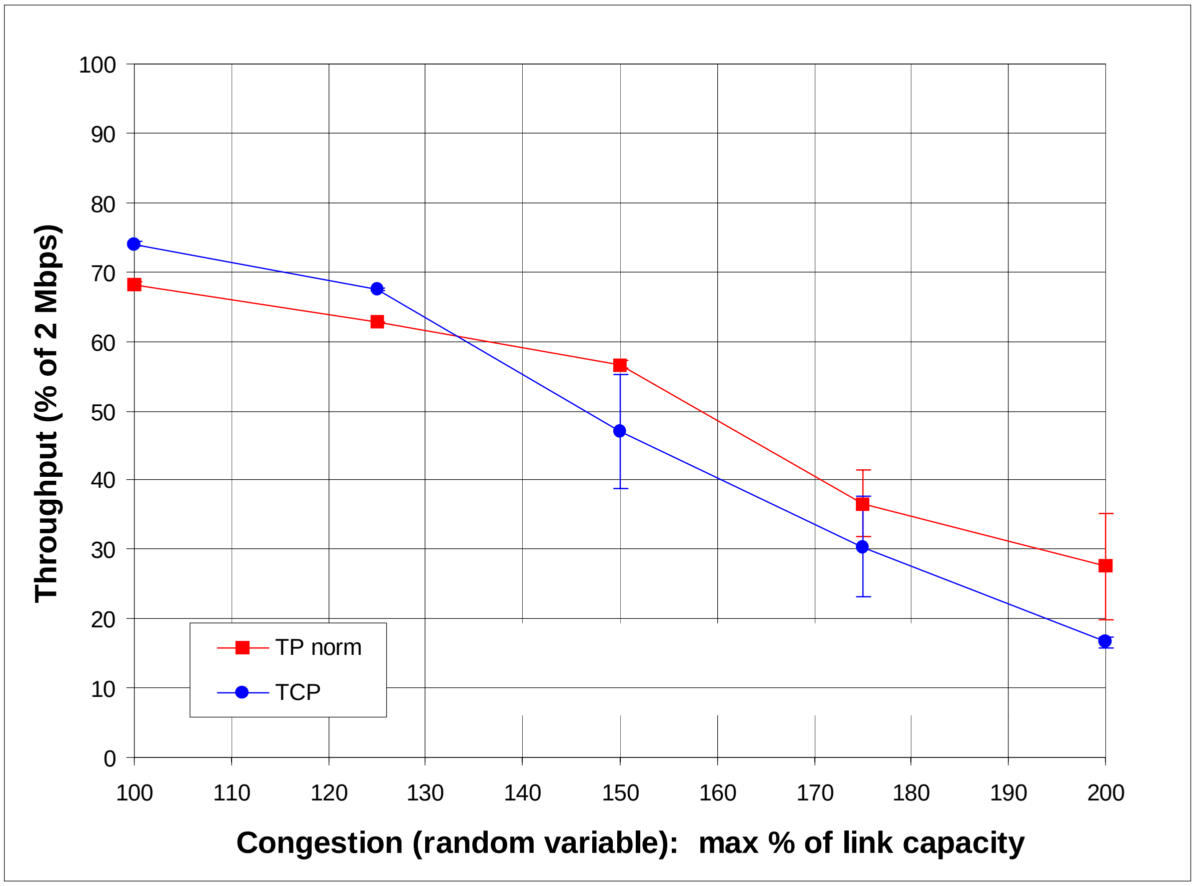


Figure 4. 2 Mbps Corruption Throughput (4 Mbyte file, 1400 byte packets)

If corruption is experienced on the satellite link and very little network congestion is present on the ground links, then TP can respond to errors as if they were a result of corruption (for the purposes of this report, we have defined a link to be corrupted if the BER is greater than $1x10^{-8}$). This can be done dynamically by signaling from NP if it is present, or it can be set by the network manager for a specific static scenario. For the 2 Mbps data rate and large packet size, the performance gain of TP over TCP can be seen in Figure 4 by comparing the two solid curves (TP's congestion control algorithm is turned off). For this particular set of data, window scaling was implemented for both SCPS TP and TCP and set to an appropriate value given the bandwidth-delay product of the link. We wanted to give TCP this benefit in the comparison even though many commercial implementations of TCP do not include the window scaling option. With this option, the throughput of TP is approximately 93% versus 74% for TCP at low BER (mostly due to TCP's slow-start algorithm). This performance difference would have been much greater if TCP window scaling was not implemented because more time would be spent waiting for acknowledgments with small segments in transit instead of transmitting data.

As seen in Figure 4, TCP starts performing very poorly compared to TP as the BER increases beyond $1x10^{-6}$. At $1x10^{-5}$, TCP is already down to 8% throughput while TP is still at 88%. As the error rate increases on the link, TCP responds as if the errors were due to congestion, and it reduces its transmission rate in an attempt to control the (perceived) congestion. Not only does TP avoid this response, it also implements a selective negative acknowledgment scheme to signal missing packets more efficiently, so that retransmissions may be made promptly.

A user/network manager may want to turn on congestion control depending on the scenario, and the dashed curve in Figure 4 indicates the expected performance of TP (in this corruption environment) when it is set to be able to respond to congestion also. Here, TP performs only slightly worse than TCP at low BER (due to the differences in the congestion control algorithms), but it clearly outperforms TCP at $1x10^{-6}$, and higher, error rates.

Throughput data for these tests was actually derived by measuring the delays of the packets from source to destination and knowing the amount of data transmitted. For example, Figure 5 displays the overall network delay experienced for these two protocols which resulted in the throughput summaries in Figure 4. For a 4 Mbyte file and 1400 byte packets over a virtually error-free 2 Mbps geosynchronous SATCOM link, using SCPS TP will result in a delay of approximately 16.8 seconds, whereas using TCP will result in roughly 21.1 seconds. This may or may not be significant to users, but as we will see shortly, the difference becomes more significant with smaller packet size. However, for these large packets, the delay becomes noticeably worse for TCP (Figure 5) as the BER increases past $1x10^{-6}$. By $5x10^{-5}$, the delay for TCP is nearly 450 seconds while TP is resulting in only a 21.7 second delay.

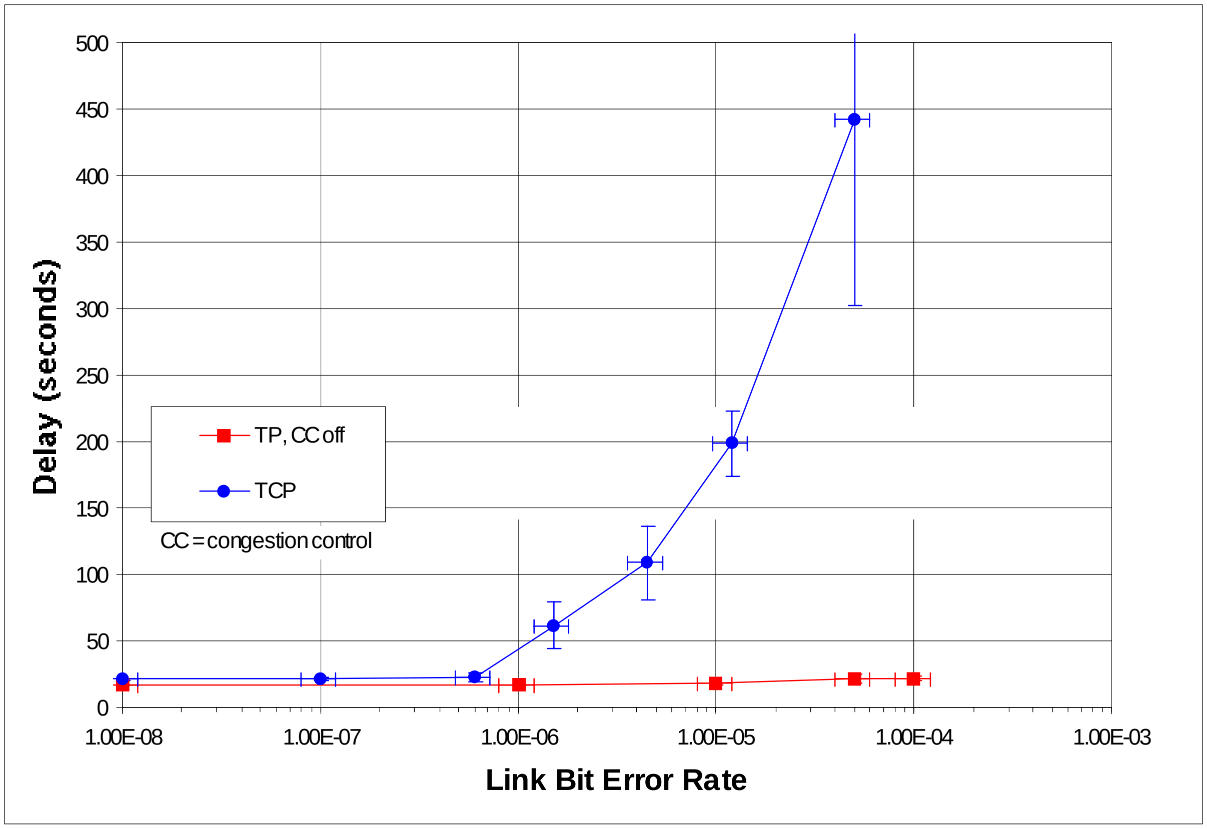


Figure 5. 2 Mbps Corruption Delay (4 Mbyte file, 1400 byte packets)

As the packet size gets smaller for this relatively high data rate geosynchronous link, the difference between SCPS TP and TCP end-to-end performance becomes more dramatic. For example, Figures 6 and 7 depict throughput and delay, respectively, for these two protocols under the same conditions as the previous data with the exception that now the packet size is reduced to 512 bytes. As seen in Figure 6, the throughput of TCP has dropped all the way down to 40% even on an error-free link. This, again, is due primarily to the TCP slow-start algorithm operating on a very large bandwidth-delay product link. SCPS TP, on the other hand, is performing nearly as well as when the packets were 1400 bytes. As the BER increases past $1x10^{-5}$, SCPS TP still maintains very good throughput, while TCP drops down into 2-8% range as it did with larger packets.

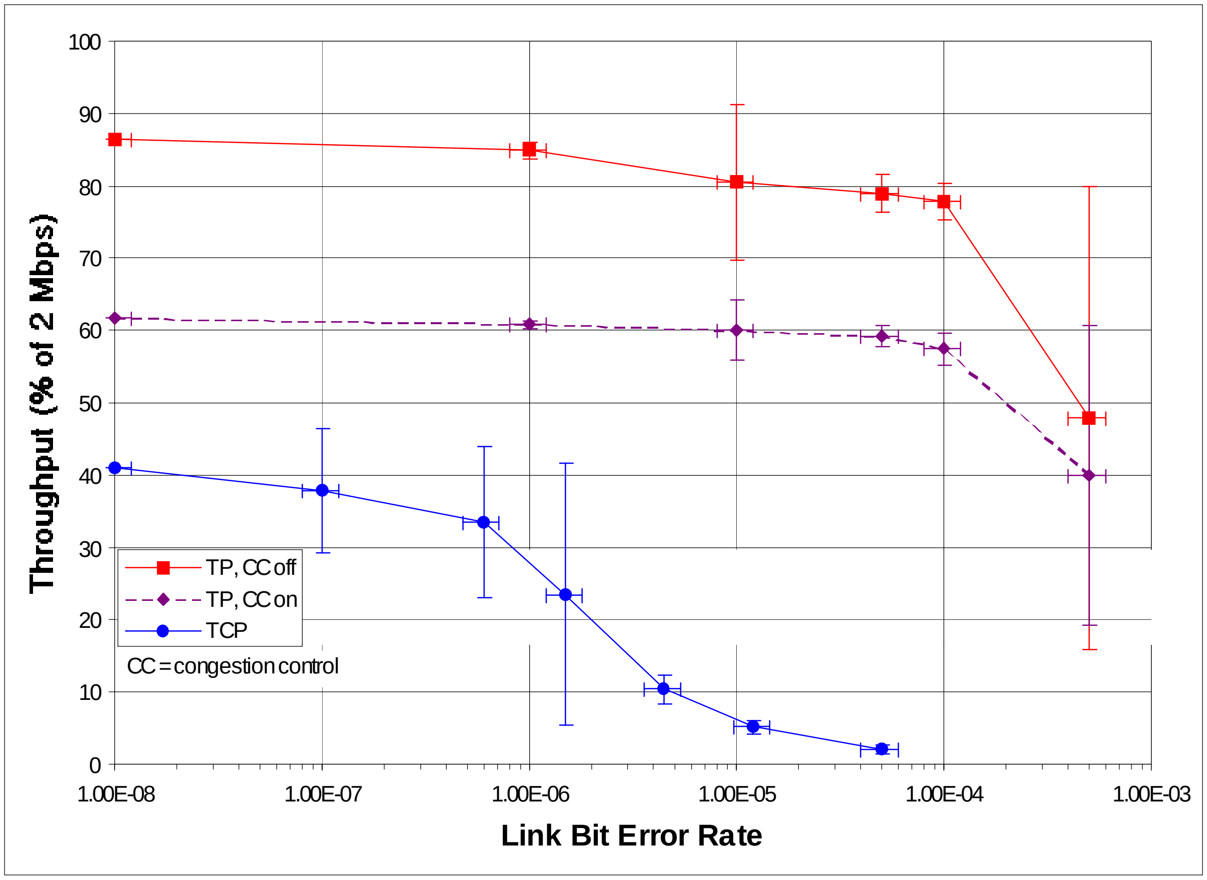


Figure 6. 2 Mbps Corruption Throughput (4 Mbyte file, 512 byte packets)

Similar end-to-end delay performance is displayed in Figure 7. Starting with no bit errors on the link, the delay experienced by SCPS TP is approximately 18.1 seconds compared to 38.1 seconds for TCP. As the BER increases, the delay with TCP increases very rapidly, but the delay for TP remains relatively flat. In fact, the performance of TCP with this packet size degrades enough at error rates higher than $1x10^{-5}$, that it becomes difficult to even close a connection.

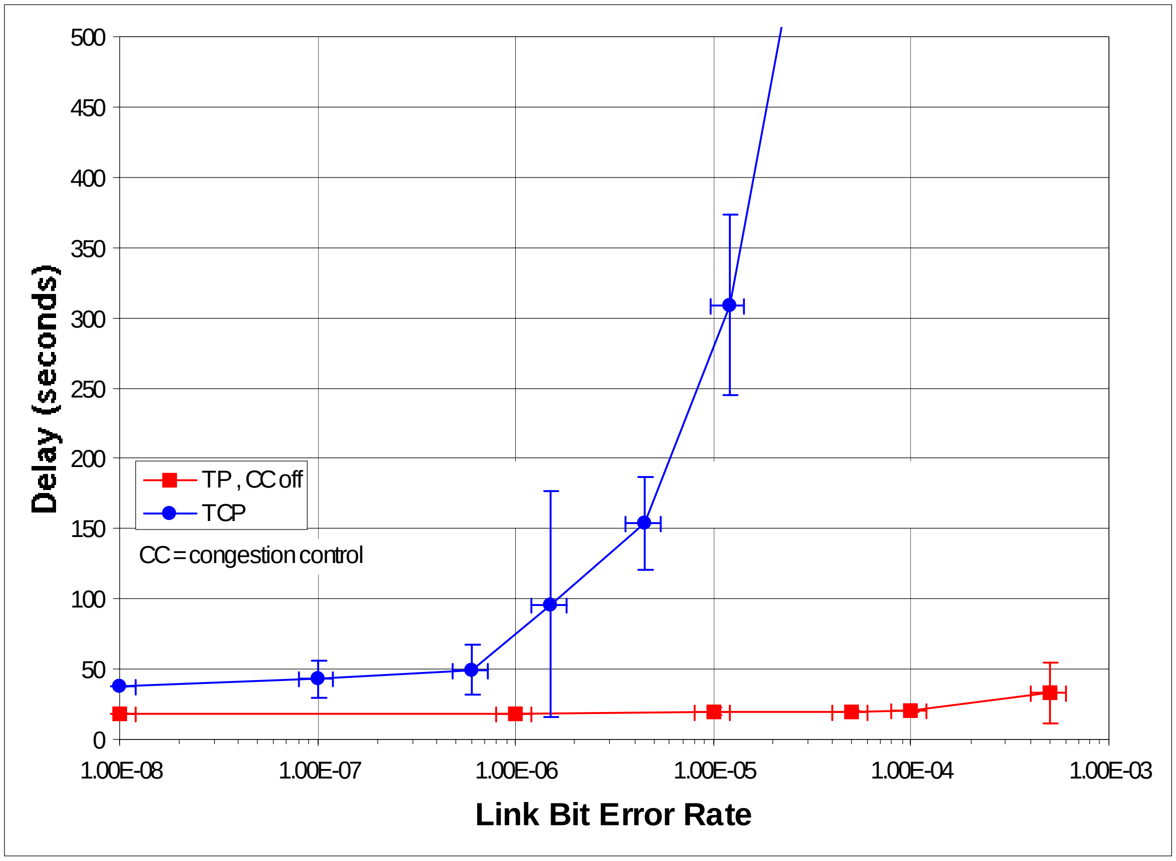


Figure 7. 2 Mbps Corruption Delay (4 Mbyte file, 512 byte packets)

The general trend in the performance differences between SCPS TP and TCP continues as the packet size is reduced further. Figures 8 and 9 depict throughput and delay, respectively, for these two protocols on a 2 Mbps link with a .5 Mbyte file and 50 byte packets. Here, the file size is reduced to be more realistic with a much smaller packet size. As seen in Figure 8, TCP throughput starts out at less than 10% even with no errors on the link. This is to be expected because a smaller amount of data is initially transit as the protocol is waiting on an acknowledgment (due to the slow-start algorithm). SCPS TP also suffers on this high data rate link with small files and small packets (just over 25% throughput with no bit errors), but it is still over 2.5 times better than TCP for these same conditions. As the BER increases, TCP performance degrades as before, but this time the curve looks much flatter because it starts so poorly at lower error rates. However, TP maintains its performance very well as the link error rate increases.

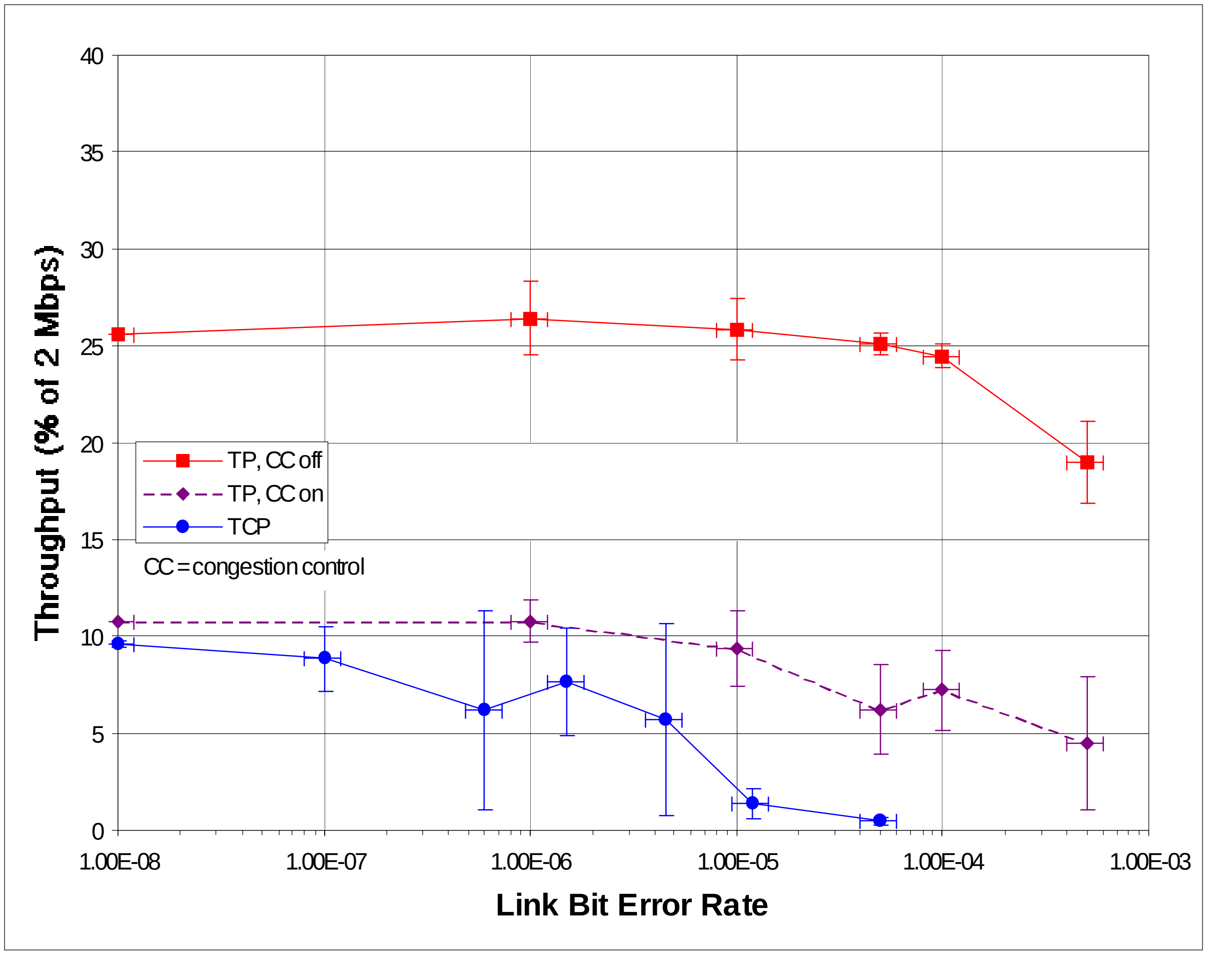


Figure 8. 2 Mbps Corruption Throughput (.5 Mbyte file, 50 byte packets)

End-to-end delay performance is displayed in Figure 9 for the 50 byte packets and the .5 Mbyte file. Starting with no bit errors on the link, the delay experience by SCPS TP is approximately 7.6 seconds compared to 20 seconds for TCP. As the BER increases, the delay with TCP increases very rapidly, but the delay for TP remains relatively flat.

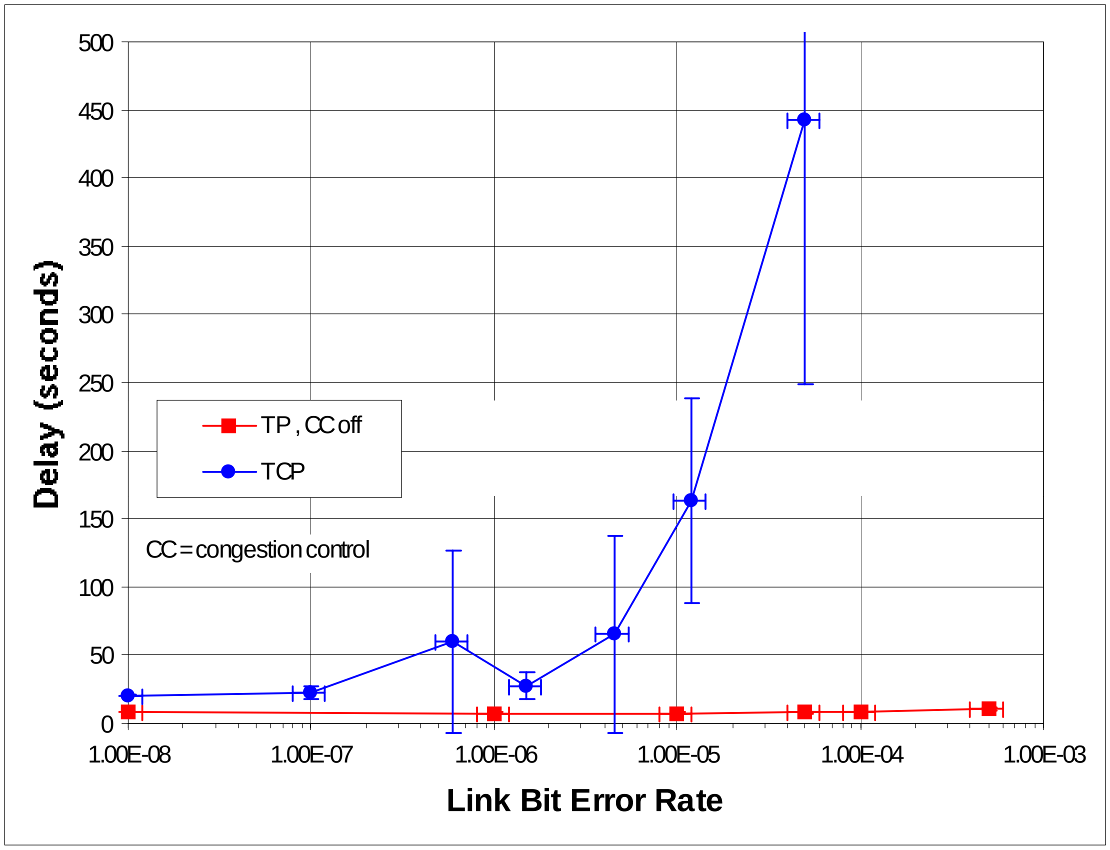


Figure 9. 2 Mbps Corruption Delay (.5 Mbyte file, 50 byte packets)

3.1.5.2 Corruption Testing - Low Data Rate

The low data rate selected for these tests was 9600 bps [1]. At this data rate, the performance of SCPS TP was compared to that of TCP at two packet sizes. The results of this comparison indicate a somewhat different story than for high data rate SATCOM links. Figure 10 indicates the relative throughput performance of these two protocols when attempting to send a medium size file (.5 Mbyte) over this 9600 bps geosynchronous link using 512 byte packets. As in the previous section, throughput is expressed as a percentage of the maximum link capacity available. Therefore, 100% throughput is equivalent to 9600 bps.

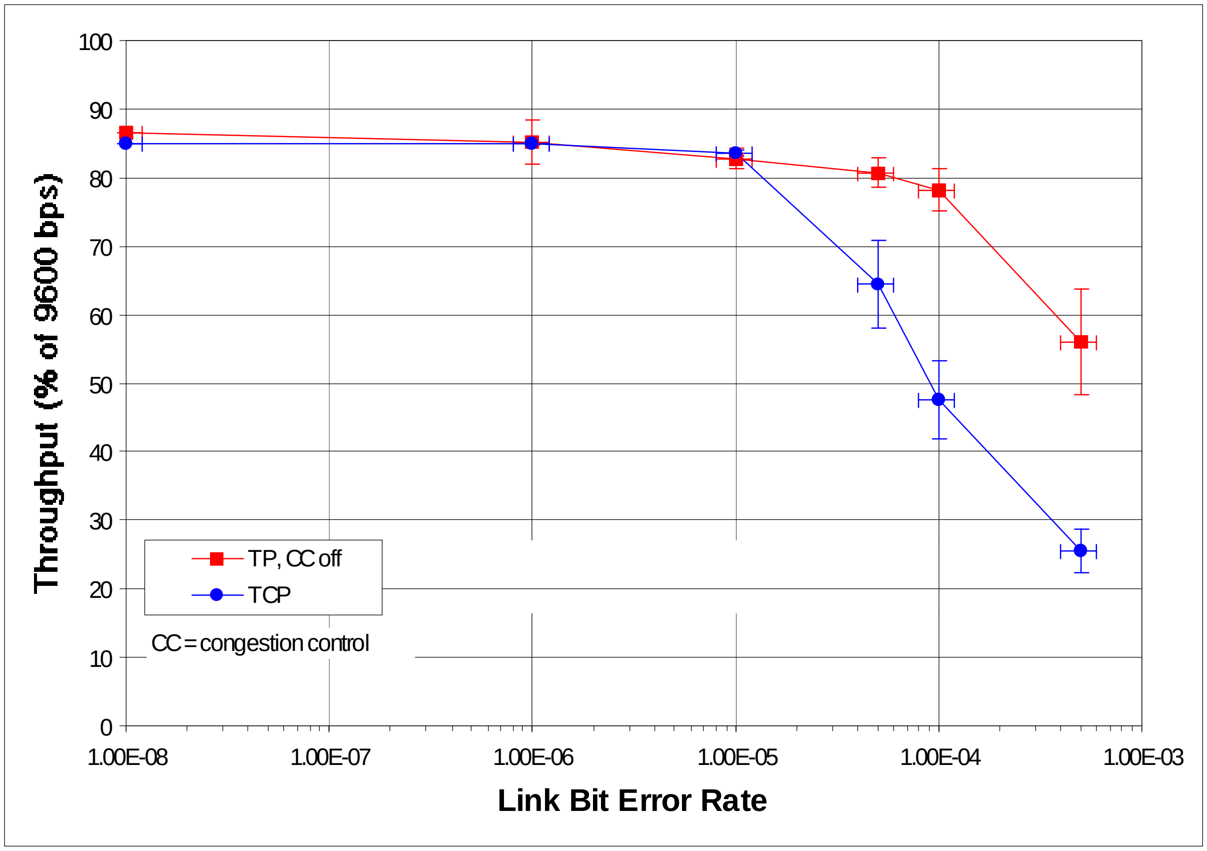


Figure 10. 9600 bps Corruption Throughput (.5 Mbyte file, 512 byte packets)

As seen in Figure 10, the throughput performance of both protocols is nearly identical (within statistical significance) when no bit errors are present on the link. The relative performance is the same for SCPS TP and TCP all the way out to $1x10^{-5}$ BER. It is not until a BER of $1x10^{-4}$ that a significant difference in performance is seen. But even then, a small increase in signal level on the link (less than .5 dB increase would decrease the BER from $1x10^{-4}$ to $1x10^{-5}$) would bring the throughput performance of TCP back up to that of TP. Note that this would be true of any link that incorporates rate 1/2, constraint length 7, convolutional coding (as most DOD SATCOM links do).

The relative end-to-end delay performance under these conditions is depicted in Figure 11. As expected based on the throughput results, the delay of TP and TCP is nearly identical from very low BER all the way out to $1x10^{-5}$. Then, the delay experienced by using TCP starts increasing more rapidly for TCP than for TP.

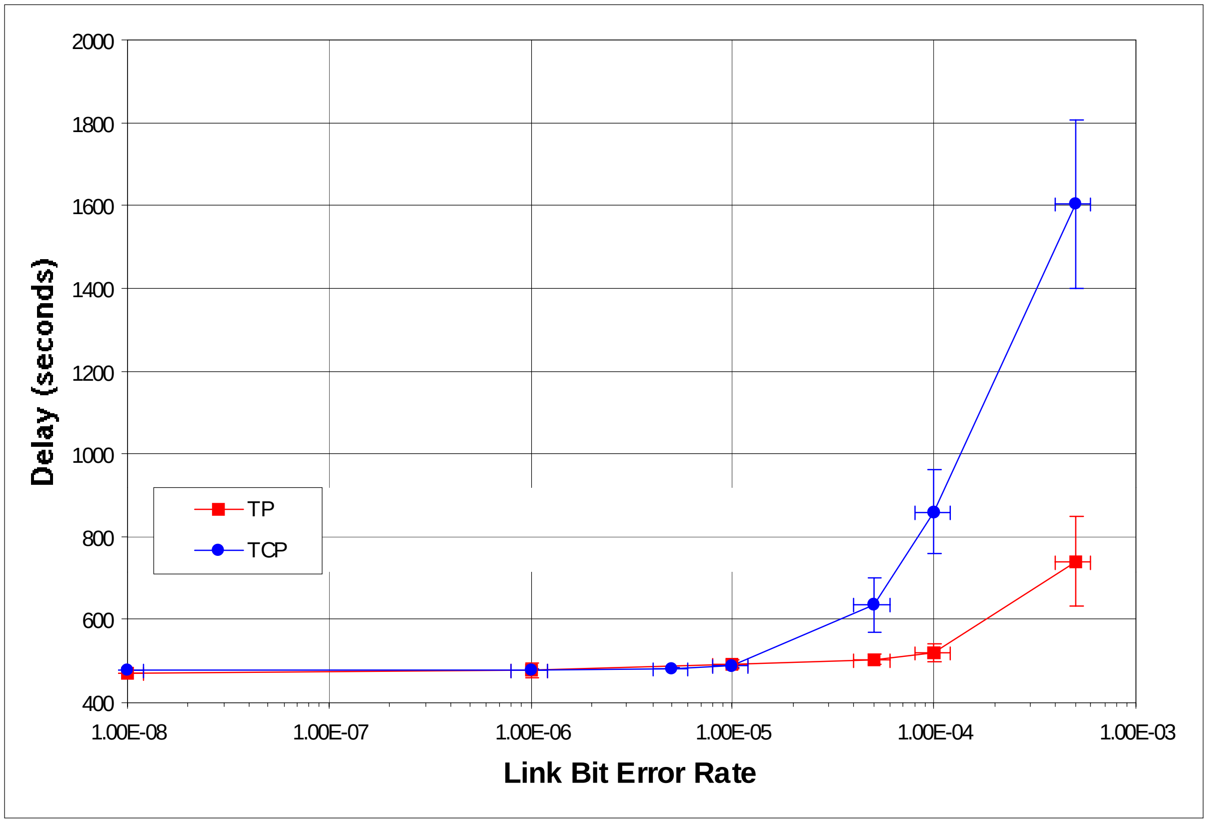


Figure 11. 9600 bps Corruption Delay (.5 Mbyte file, 512 byte packets)

When the packet size is reduced to 50 bytes on this relatively low data rate geosynchronous link, the results of our testing indicate only a slight advantage in end-to-end performance of TP, even at low BER. Figures 12 and 13 depict throughput and delay, respectively, for these two protocols on a 9600 bps link for a .5 Mbyte file using 50 byte packets. As seen in Figure 12, the throughput of TCP is in the high 40s (percent) from low BER all the way out to $1x10^{-5}$. TP maintains a throughput of approximately 54% from low BER out to $1x10^{-5}$.

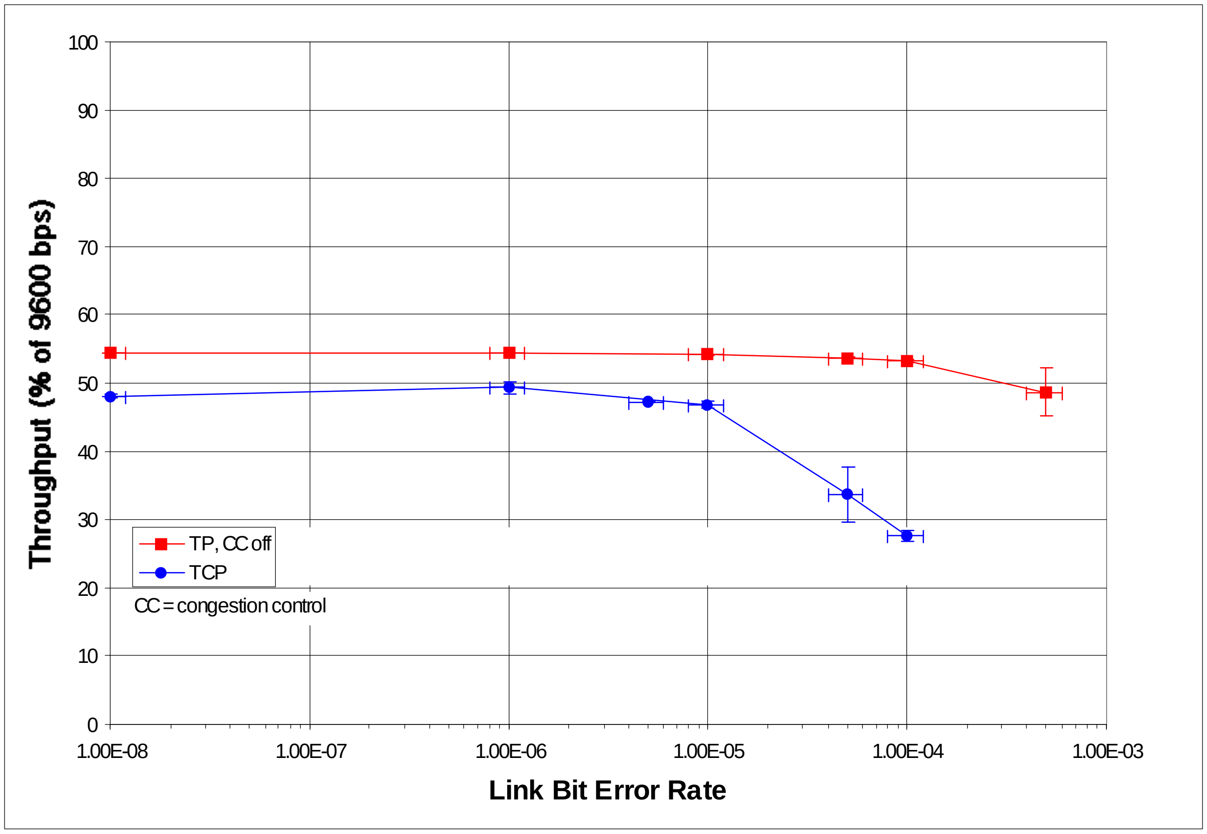


Figure 12. 9600 bps Corruption Throughput (.5 Mbyte file, 50 byte packets)

The relative end-to-end delay performance of TP for low data rate and 50 byte packets has a slight advantage over TCP at low BER, as depicted in Figure 13. As expected based on the throughput results, the delay experienced by using TCP starts increasing more rapidly for TCP than for TP at error rates on the link higher than $1x10^{-5}$.

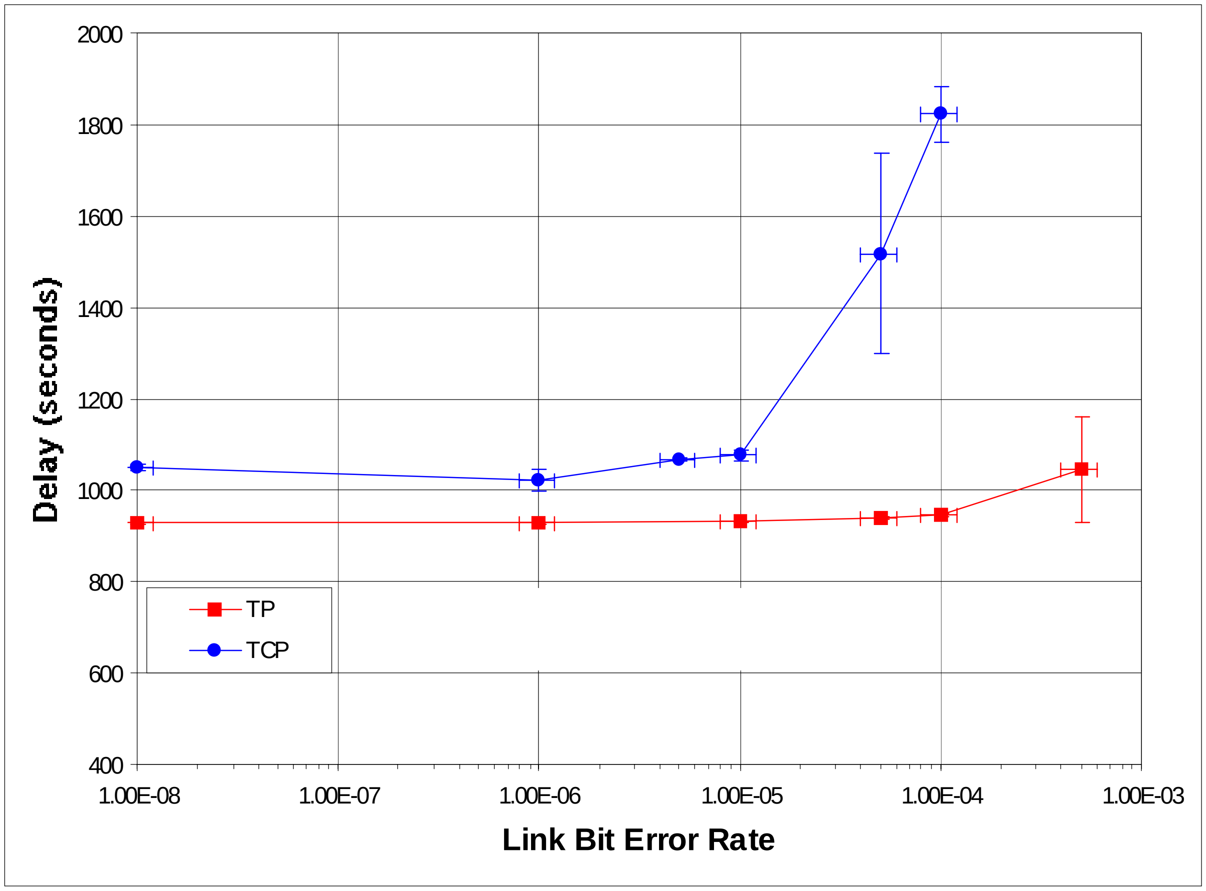


Figure 13. 9600 bps Corruption Delay (.5 Mbyte file, 50 byte packets)

3.1.5.3 Congestion Testing - High Data Rate

For congestion testing, the primary independent variable is the level of congestion present in the network. For these tests, throughput and delay were measured as a function of network congestion for two different data rates, three packet sizes, and two file sizes.

The high data rate selected for these tests was 2 Mbps [1]. At this data rate, the congestion performance of SCPS TP was compared to that of TCP at three packet sizes. Figure 14 indicates the relative throughput performance of these two protocols when attempting to send a relatively large file (4 Mbyte) over this geosynchronous link using large packets (1400 bytes) appropriately sized for the file. As in previous figures, throughput is expressed as a percentage of the maximum link capacity available. Therefore, 100% throughput is equivalent to 2 Mbps.

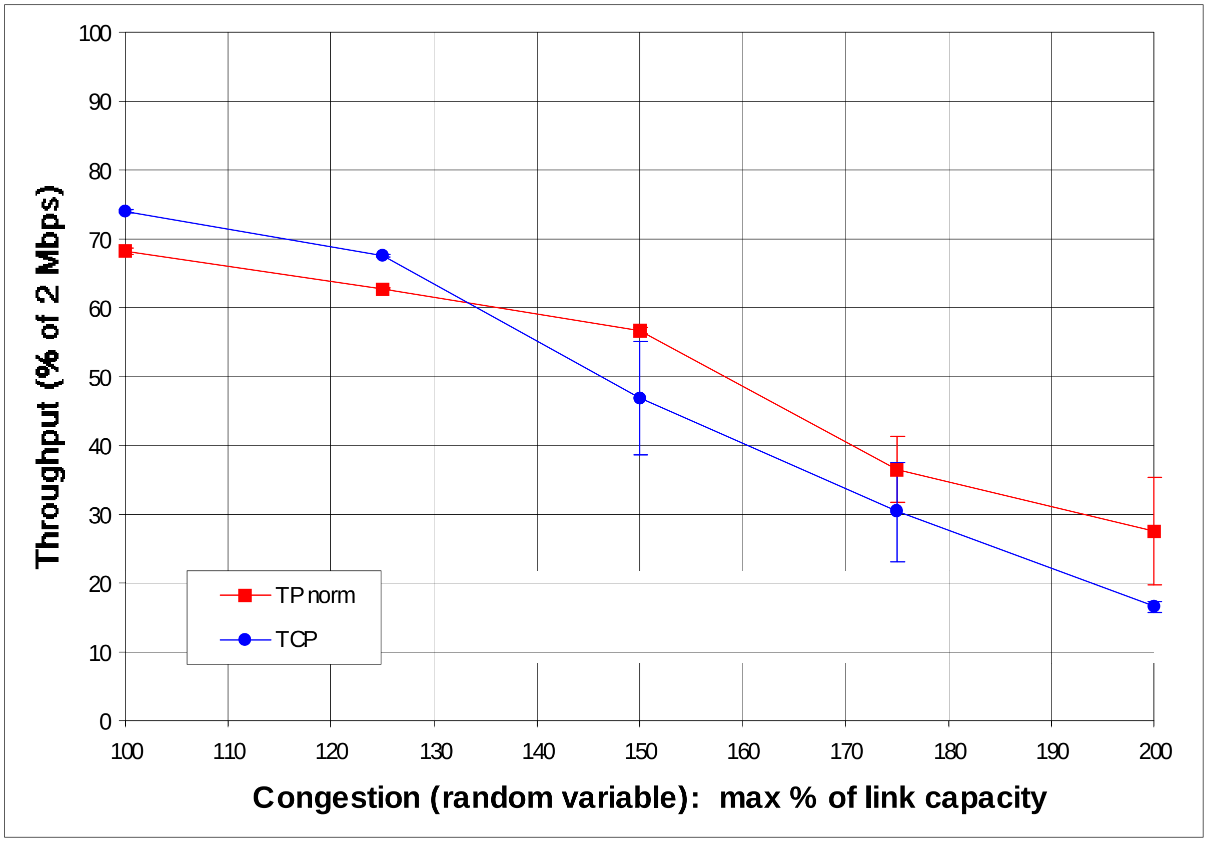


Figure 14. 2 Mbps Congestion Throughput (4 Mbyte file, 1400 byte packets)

As stated earlier, the data source (see Figure 3) for these tests always generated data at a rate matched to the capacity of the SATCOM link. However, depending on the transport protocol being used, the instantaneous traffic offered on the connection may have actually been greater than the capacity of the link. In addition, the congestion traffic generator (WS3 in Figure 3) generated random congestion traffic uniformly distributed between zero and a maximum specified by the tester. The aggregate traffic presented to the router in this figure was a combination of the traffic from these two sources. For example, if a maximum of 100% was specified for a 2 Mbps test, this meant that no congestion traffic was generated. If a maximum of 200% was specified, this meant that the congestion source was generating random traffic uniformly distributed between 0 and 2 Mbps.

From Figure 14, it appears as though TCP starts out performing slightly better than TP at no congestion (which is consistent with corruption test results when congestion control is enabled for TP), then the performance of TCP crosses over and remains slightly worse than TP as the congestion level increases. Some of this behavior can be explained by the differences in the congestion control algorithms used by the two protocols, which is discussed more in Appendix A. The TCP implementation used for this test employs the standard TCP congestion control algorithm (Van Jacobson), while SCPS TP uses a modified version of TCP-Vegas. Also, it should be noted that given the 95% confidence intervals for this data, some of this trend may not be statistically significant, although it is consistent for congestion levels higher than 150% in the figure.

The results for 512 byte packets are shown in Figure 15. Here, TP starts out better than TCP with no congestion (again, consistent with corruption testing), then crosses over the TCP performance to become slightly worse at higher congestion levels. Again, it is not clear that this is statistically significant, given the large variance in the data. This large variance was partly due to the way the congestion traffic was generated. In some cases, the test duration may not have been long enough to allow the traffic generator to step through enough values to make the distribution look uniform.

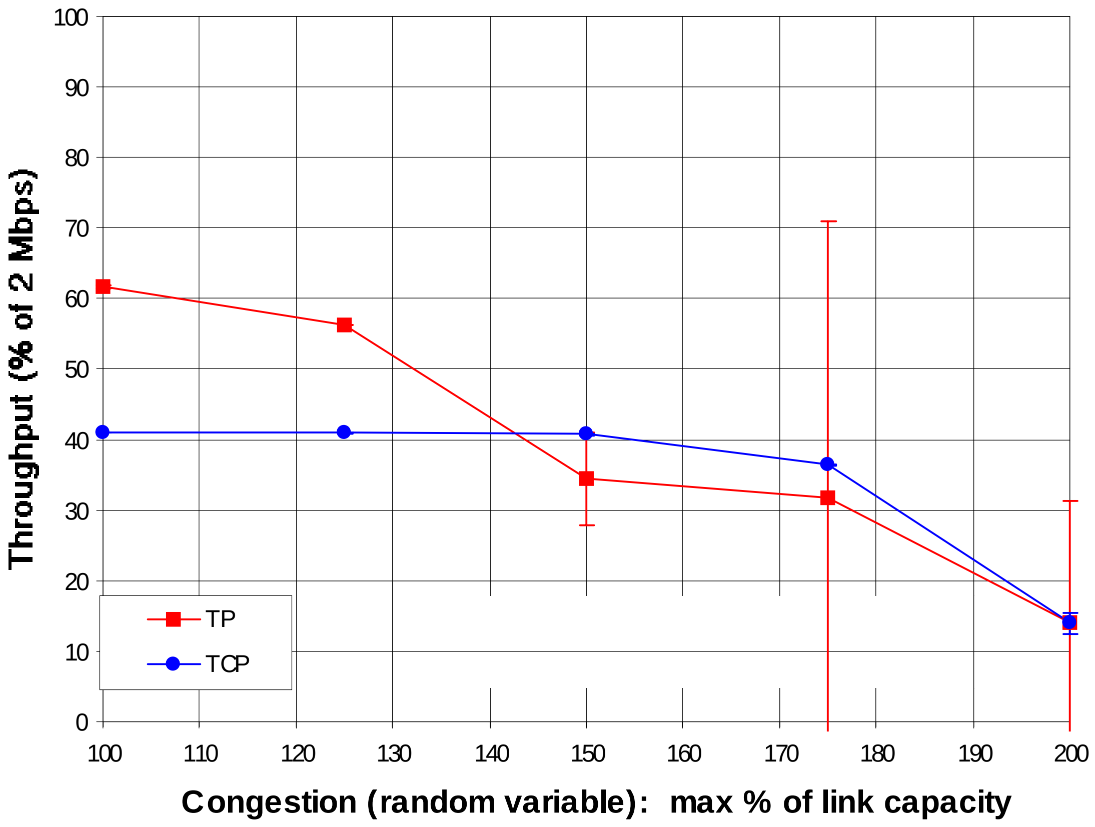


Figure 15. 2 Mbps Congestion Throughput (4 Mbyte file, 512 byte packets)

The results for 50 byte packets are shown in Figure 16. Here, TP crosses over the TCP performance a number of times, but it is not likely that this is statistically significant, given the large variance in the data. Taking into account this variance, the performance of each protocol seems very similar.

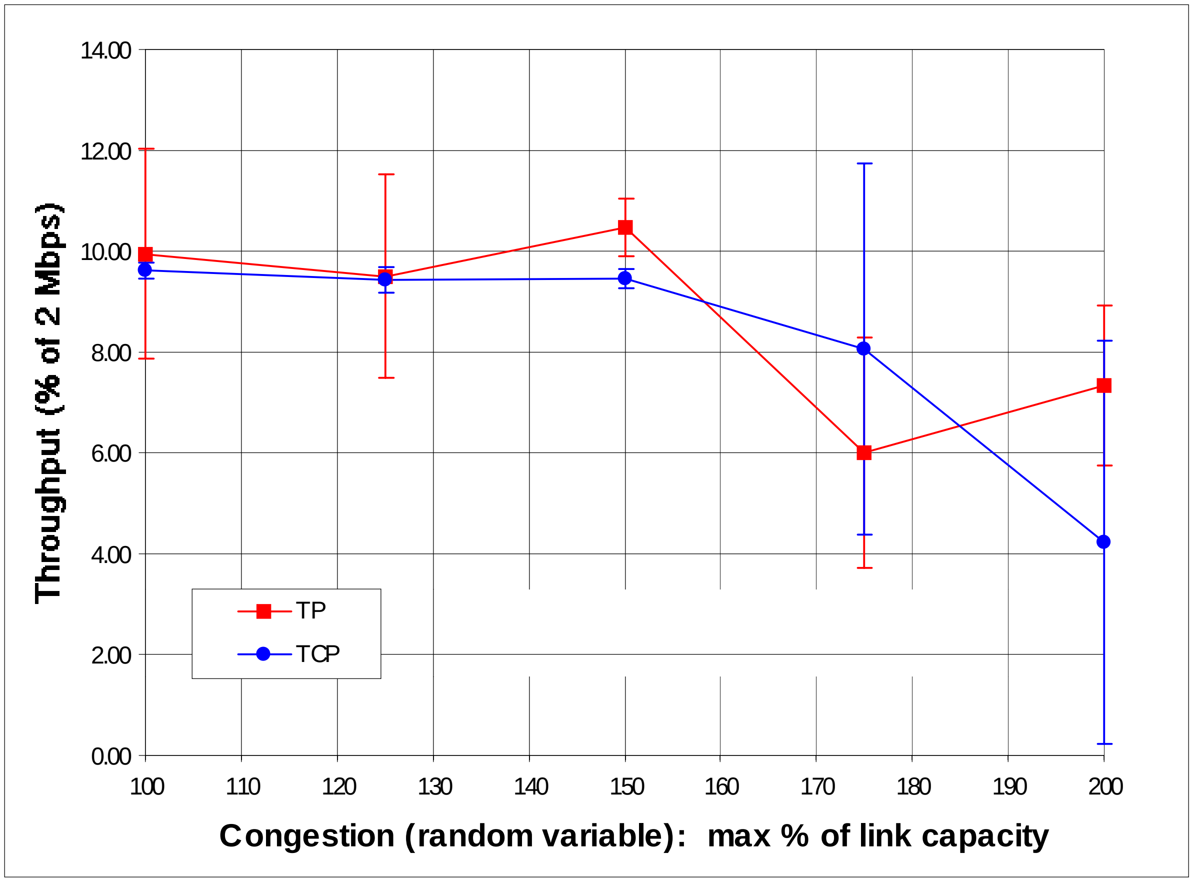


Figure 16. 2 Mbps Congestion Throughput (.5 Mbyte file, 50 byte packets)

3.1.5.4 Congestion Testing - Low Data Rate

The low data rate selected for these tests was 9600 bps [1]. At this data rate, the congestion performance of SCPS TP was compared to that of TCP at two packet sizes. Figure 17 indicates the relative throughput performance of these two protocols when attempting to send a medium size file (.5 Mbyte) over this 9600 bps geosynchronous link using 512 byte packets. As in previous figures, throughput is expressed as a percentage of the maximum link capacity available. Therefore, 100% throughput is equivalent to 9600 bps.

As seen in Figure 17, the performance of these two protocols in the congestion environment under these conditions is nearly identical. Furthermore, the variance in the data was almost nonexistent.

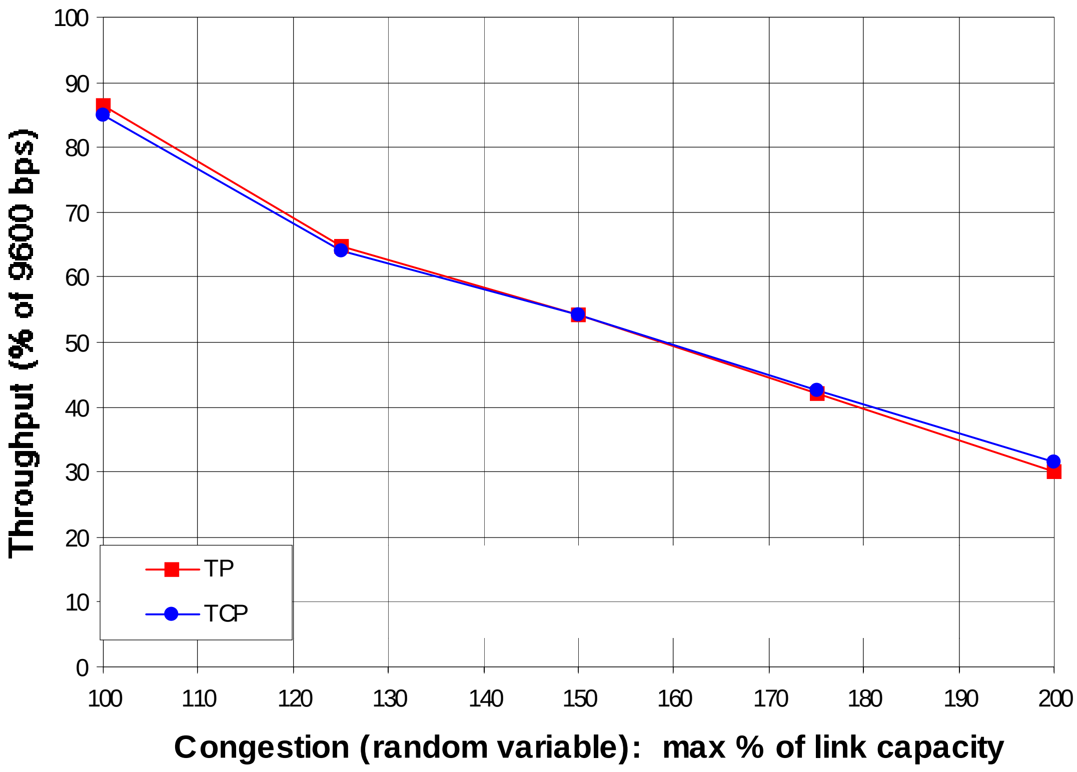


Figure 17. 9600 bps Congestion Throughput (.5 Mbyte file, 512 byte packets)

The results for 50 byte packets are shown in Figure 18. TP indicates a slight advantage over TCP for this packet size. Data was difficult to obtain for TCP beyond 150% congestion and for TP beyond 175% congestion because the connection would time out before the data transfer could be completed.

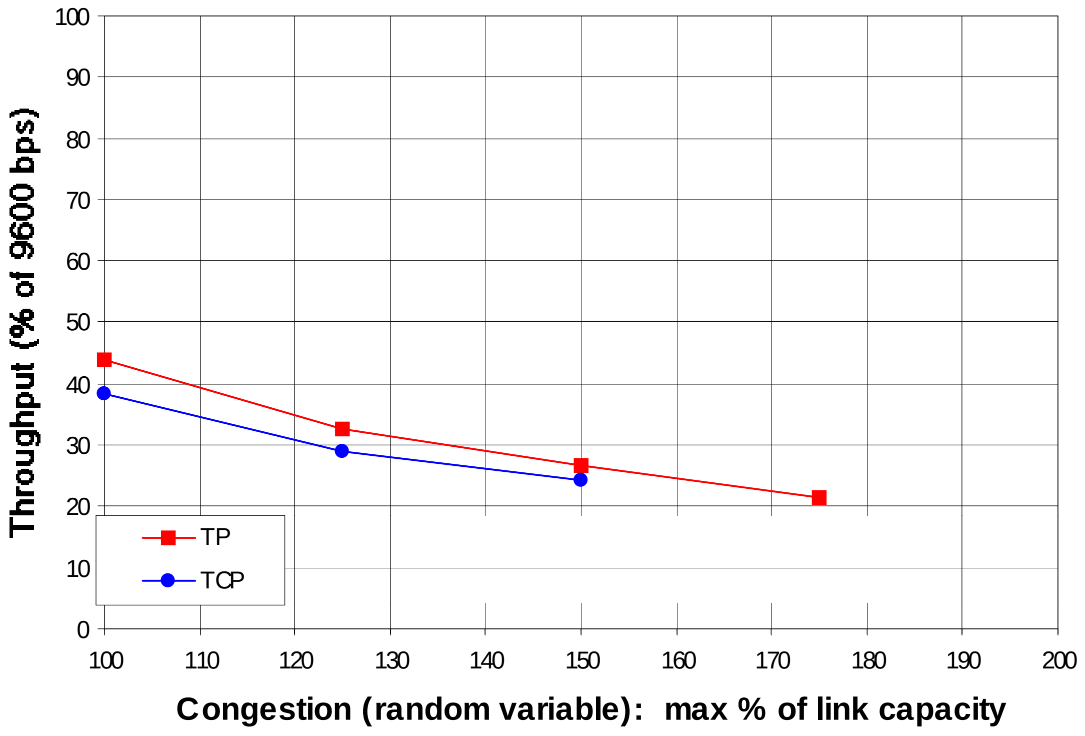


Figure 18. 9600 bps Congestion Throughput (.5 Mbyte file, 50 byte packets)

3.2 File Handling Protocol

SCPS FP testing was the second highest priority in FY97 in support of potential SATCOM users. SCPS FP may be desirable to a SATCOM user if that user has a need to transfer files or individual records of files and may want the protocol to automatically resume the transfer after it is interrupted. Because not all SATCOM users will have these needs, this FP functionality was partially demonstrated; however, detailed performance testing with comparison to commercial FTP was not conducted.

3.2.1 Protocol Requirements

Table 3 identifies those functional requirements of FP that were planned to be tested in the SATCOM environment. These are derived from [3].

Table 3. SCPS File Handling Protocol Requirements

Ref Para	Requirement	Type of Test
F.2	Operations on file records.	N/A
F.2.2	Shall provide the capability to insert a record or set of records into any location in a file.	F
F.2.3	Shall provide the capability replace any record or set of records within a file.	F
F.2.4	Shall provide the capability to delete any record or set of records within a file.	F
F.3	Shall provide the capability for either of two end systems to send and receive a complete file.	F
F.5	User initiated interrupt and abort.	N/A
F.5.1	Shall provide the capability for the user to cause an interrupt of a file transfer after the start of the transfer.	F
F.5.2	Shall provide the capability for a user to terminate a file transfer after the start of the transfer.	F
F.6	System-detected interrupt notification.	N/A
F.6.1	Shall recognize a notification that the communications supporting a file transfer has been interrupted.	F
F.7	Resumption after interrupt	N/A
F.7.1	Shall provide the capability to manually resume a file transfer from the point of interruption.	F
F.7.2	Shall provide the capability to automatically resume a file transfer from the point of interruption.	F

F=functional test, P=performance test

3.2.2 General Experiment Design

All FP functional testing was conducted under the corruption environment identified for TP testing. FP was an application running on top of TP, which provided a transport service for these tests. For each experiment, files and records of files to be sent across the satellite link to the other end application were generated with a utility program. In order to conserve test resources, a selected portion of basic FP functions (those identified in Table 3) were verified under a single link condition (no bit errors) and in the forward direction only. These basic functions are believed to be of benefit to potential SATCOM users compared to the capabilities of commercially available FTP.

3.2.3 Test Method

The specific network configuration used for FP functional testing is identified in Figure 3. This is identical to the configuration for TP testing, except FP also resided at the two end workstations (WS1 and WS2). During these tests, WS3 remained connected to the LAN, but did not generate any congestion traffic. Tests were executed and data was collected both locally at Rome Laboratory and remotely from Reston, VA as indicated in Figure 3.

For basic file transfer and record update operations, the test was initiated by requesting the operation at the FP client on WS1. On record updates, the functionality was demonstrated with records of 512 bytes, then with records of 2, 8, and 32 kbytes. On file transfers, the functionality was demonstrated with files of 8, 32, 256, and 1024 kbytes.

For manual and automatic file interrupt operations, a large file transfer was initiated from the FP client on WS1 over a low data rate link. For the manual function, the intent was to demonstrate that the transfer could be intentionally interrupted and restarted from the FP client. For the automatic function, the intent was to simulate a link outage by disabling the physical layer modem, then demonstrate that FP automatically completed the file transfer when the modem was enabled.

3.2.4 Data Reduction

The data reduction was minimal for this test. At the end of each file or record transfer, we simply verified that the application acknowledged completion of the transfer, then we used a utility to compare the file received to the file transmitted to verify they were identical.

3.2.5 Summary of Results

All record update and file transfer operations were completed successfully.

Manual interrupt/restart and automatic interrupt/restart functions were never completed successfully. FP was implemented through a socket interface for this testing. A problem was discovered with this interface in the course of this testing, but we did not have sufficient resources on this test program to resolve the problem. As a result, we did not complete the demonstration of these features.

3.3 Network Protocol

Historically, military SATCOM has been used to connect end users who are collocated with terminals and not connected to networks. However, a number of scenarios have been identified in which it is desirable to connect an existing ground network (possibly TCP/IP-based) to a remote section of a network through a SATCOM link, as depicted in Figure 2.

Many near-term potential SATCOM users may not be willing to implement SCPS protocols at all the ground nodes in Figure 2. One way to support those users without much impact to existing systems is to implement SCPS (or a subset of SCPS) only at the end points of the network and at the two nodes connected to the satellite link. In this scenario, SCPS (TP and NP at a minimum) is installed at the end node in each network and is encapsulated by IP. The commercial IP then routes information through the ground network until it reaches the satellite link. Here, TP would interface with NP to deal with issues pertaining to the physical satellite link. In this way, true end-end reliability is provided via SCPS without having to change existing intermediate ground nodes in the network. The test configuration that supported this environment is identified in Figure 19.

As discussed earlier, SCPS TP testing was the highest priority in FY97 because it provides the largest potential benefit in support of near-term SATCOM users. However, NP can also provide benefit in the environment just described, mostly through its capability to signal to TP the presence of both corruption and congestion and the capability to assign precedence to individual network packets. Therefore, tests were conducted in FY97 to demonstrate these two functions of NP in the congestion and corruption environments.

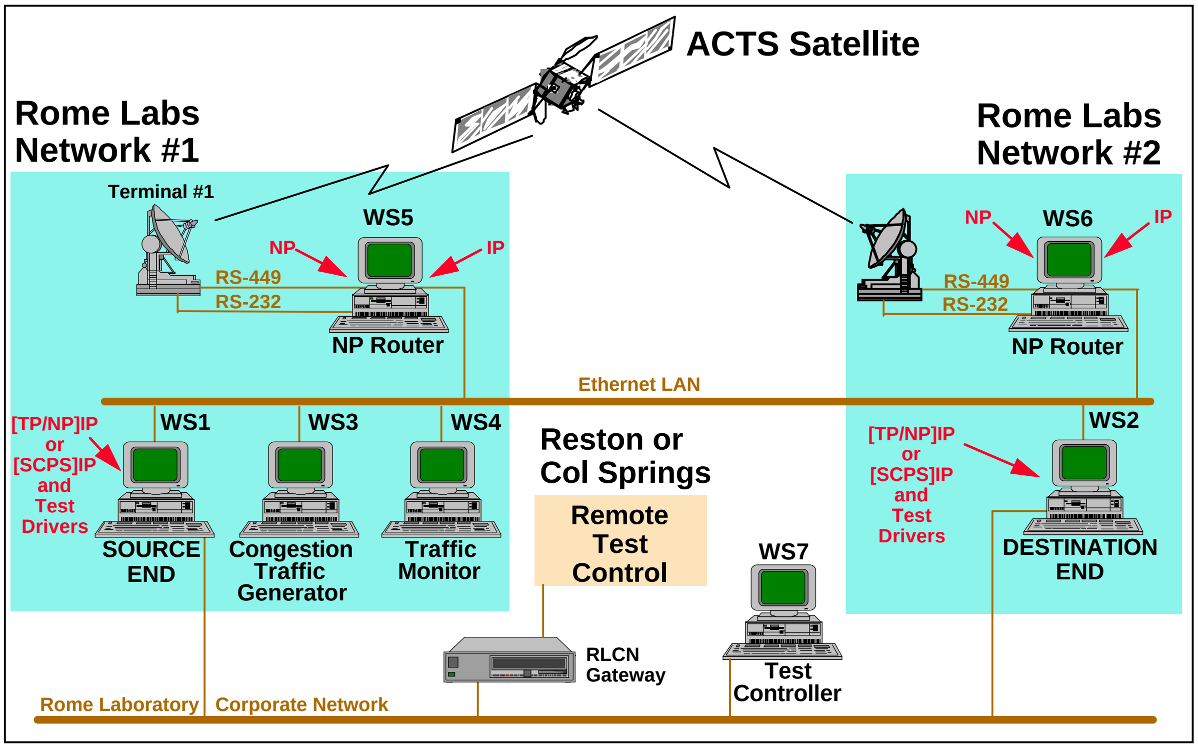


Figure 19. Test Configuration 2

3.3.1 Protocol Requirements

Table 4 identifies those functional requirements of NP that were tested in the SATCOM environment. These are derived from [3].

Table 4. SCPS Network Protocol Requirements

Ref Para	Requirement	Type of Testing
N.4	Separate reporting for congestion and corruption.	N/A
N.4.1	Shall be able to detect and differentiate between network congestion and network data corruption.	F
N.4.2	Shall be able to report each of these two conditions to the transport protocol in a way that differentiates between them.	F
N.4.3	Shall be able to manage and possibly discard data in response to congestion.	F
N.4.4	Shall be able to discard data in order from lowest to highest precedence.	F
N.5	Support for precedence handling.	N/A
N.5.1	Shall be able to recognize the precedence level specified by the application.	F
N.5.2	Shall be able to provide a default precedence level for those packets that require one.	F
N.5.3	Shall be able to assign the proper precedence level to each outgoing packet that requires one.	F
N.5.4	Shall be able to recognize the precedence level associated with an incoming packet.	F
N.5.5	Shall be able to process incoming packets in accordance with their assigned precedence level.	F
N.5.6	Shall provide the capability for system configuration personnel to set the default precedence level for a system.	F

F=functional test, P=performance test

3.3.2 General Experiment Design

All NP functional testing was conducted under separate corruption and congestion environments. First, the ability of NP to signal corruption to TP was tested on a corrupted satellite link without ground network congestion. Second, the ability of NP to signal congestion to TP was tested in a congested ground network without satellite link corruption. Finally, the ability of NP to properly deal with various levels of precedence was demonstrated in the presence of ground network congestion.

3.3.3 Test Method

The specific network configuration for NP functional testing, which is identified in Figure 19, is similar to the configuration for TP testing. The primary difference is that NP also resided at the two end workstations and two additional workstations were needed to act as NP routers at the satellite terminals.

For corruption environment tests, WS1 hosted [TP/NP]IP and TP test drivers associated with the source end of the connection. WS2 hosted [TP/NP]IP and TP test drivers associated with the destination end of the connection. During these tests, WS3 did not generate any congestion traffic. WS5 and WS6 hosted IP to interact with the Ethernet on each side of the satellite link. These workstations also hosted NP which interacted with the modem and satellite terminal through a high speed RS-449 serial interface.

To demonstrate the function of NP signaling corruption to TP, the link BER was set to various levels above $1x10^{-6}$, TP connections were initiated, and it was verified that TP was signaled to respond to corruption. For additional engineering information, throughput and delay were measured similarly to TP performance testing.

For congestion environment tests, WS3 generated traffic to produce various levels of congestion to the NP router at WS5. To demonstrate the function of NP signaling congestion to TP, the link BER was set to less than $1x10^{-8}$, TP connections were initiated, and it was verified that TP was signaled to respond to congestion. Delay and throughput were also measured.

To demonstrate the function of NP packet precedence, separate TP connections were established and NP packets were assigned different priority for each connection. Delay and throughput were measured to verify that packets with highest priority were delivered with minimum delay while packets with lowest priority were delivered with the longest delays.

3.3.4 Data Reduction

For the signaling tests, it was directly verified that TP was set to respond to either congestion or corruption, and no additional data reduction was necessary. For the packet precedence testing, data was collected by WS4 (see Figure 19) similar to TP testing, and test drivers performed initial data reduction in order to report throughput and delay.

3.3.5 Summary of Results

The function of NP signaling to TP in response to corruption or congestion was successfully demonstrated. The function of NP packet precedence was also successfully demonstrated by showing that the lowest priority packets always resulted in the longest delays, while the highest priority packets always resulted in the shortest end-to-end delays.

Section 4

Overall Test Summary

4.1 Transport Protocol

The performance of SCPS TP relative to TCP was evaluated separately in this test effort for link corruption and network congestion environments.

4.1.1 Corruption Environment

In the corruption environment with congestion control turned off, SCPS-TP always outperforms TCP over large bandwidth-delay product links (tested here on a 2 Mbps transponder in geosynchronous orbit). Even with no bit errors on the link, TP performs better than TCP due mostly to the slow-start congestion algorithm used with TCP. This performance improvement is significant even for relatively large packets, but it becomes substantial for smaller packets. And, the performance gains summarized herein would be much greater if TP is compared to a version of TCP that does not implement the window scaling option. As the BER on the link increases, the performance improvement of TP relative to TCP in this environment also increases (even for large packets) due mostly to TP's ability to respond to bit errors as corruption, not congestion. Even when the congestion control algorithm for TP is activated, TP still outperforms TCP in all cases except for when large packets are being transmitted and there are no bit errors on the link. For these conditions, TCP performs only marginally better than TP due to the differences in their congestion control algorithms.

When the data rate on the link is reduced substantially (9600 bps in our case), the performance gains of TP relative to TCP are not substantial. For larger file sizes and packets, the performance is nearly identical at a low BER. As the link BER increases, TP has a reasonable advantage over TCP. However, for the typical modulation and coding used on military SATCOM links, this performance gain can be neutralized by a small increase (less than 0.5 dB) in signal-to-noise ratio on the physical link. The relative performance of TP compared to TCP at this data rate is similar for small files and packets, except that TP has a slight advantage over TCP even at a low BER.

4.1.2 Congestion Environment

In the congestion environment, the current implementation of SCPS TP performs similar to TCP at the high data rate regardless of file or packet size. TCP may have a slight advantage at very low congestion levels due to the differences in the congestion control algorithms, but this seems to get reversed at higher levels of congestion.

At the lower data rate, TP and TCP appear to perform nearly identical for larger packet sizes. When a smaller packet size is used, TP appears to have a slight advantage over TCP at all levels of congestion.

4.2 File Handling Protocol

All record update and file transfer operations were successfully demonstrated, although manual interrupt/restart and automatic interrupt/restart functions were never completed. A problem was discovered with the implementation of the "sockets" programming interface in the course of this testing, and we did not have sufficient resources on this test program to resolve the problem.

4.3 Network Protocol

The function of NP signaling to TP in response to corruption or congestion was successfully demonstrated. The function of NP packet precedence was also successfully demonstrated by showing that the lowest priority packets always resulted in the longest delays, while the highest priority packets always resulted in the shortest end-to-end delays.

4.4 Security Protocol

The intent of this test effort was to functionally demonstrate a subset of SCPS SP features. However, sufficient resources did not exist to enable us to conduct these tests.

Section 5

Conclusions

This test program was implemented in order to show the utility of SCPS to existing and near-term DOD SATCOM applications. One of the biggest potential benefits of SCPS in these near-term scenarios is provided by TP. The use of this protocol can result in reduced end-to-end delays and increased throughput on corrupted links (SATCOM, in general) with large bandwidth-delay products. When the link data rate or the propagation delay is small, the performance gains are marginal. This has implications for DOD TT&C links as well because they tend to be lower data rate links. However, we can expect data rates for TT&C links to increase in the future as the demand for more capacity and more services increases. From this and previous test efforts, we know that significant improvements in throughput are obtainable using SCPS-TP on transponded geosynchronous links starting at data rates somewhere between 10 kbps and 200 kbps. Given the limited data we have, it is difficult to say at what point the performance gain is significant, and this will depend on the requirements of each user. Further testing and simulation should be conducted in conjunction with a more comprehensive assessment of potential user requirements.

SCPS FP can benefit those near-term SATCOM users who need to transfer files or individual records of files. As demonstrated in this test program, the ability to update records instead of entire files can be of great benefit in resource-constrained environments (for example, a low data rate link with a short access time). Although not successfully demonstrated in this test program, the ability to automatically restart a file transfer after it is interrupted (due to a link outage or other interruption in service) can be a significant benefit to all DOD data transfer applications. Within the constraints of each potential application, future users should consider implementing SCPS within the kernel of the operating system to avoid some of the difficulties encountered in this test program.

SCPS NP can benefit some near-term SATCOM users (depending on the scenario) by providing the capability to signal the presence of corruption or congestion to the transport layer. Probably the most significant benefit for the near-term SATCOM users is the ability to enforce packet precedence, which allows higher priority traffic to get through a congested network.

Near-term SATCOM users can also benefit from the end-to-end security services provided by SCPS SP. None of these services were demonstrated in this test program due to limited project resources.

SMC, the DOD, NASA, and ISO should continue to seek out near-term applications for SCPS as well as far-term ones. Although the FY97 SCPS DOD test program focused on near-term SATCOM applications, we should not lose sight of potential future applications of

SCPS to DOD space operations and to other potential applications, such as tactical communications. SMC is currently engaged in a study, with support from MITRE, to identify programs and classes of programs that can be expected to benefit the most (technically) from implementing some or all of the SCPS protocols.

There is also a benefit, although sometimes less tangible, of standardization. Standardization has the potential for cost savings due of commonality among systems, but this potential is not always realized. In addition, standardization can promote interoperability, which more often increases capability and sometimes reduces cost also. The four SCPS protocols are currently in process as formal military standards and as ISO standards. In addition, SCPS has been added to the current version of the Joint Technical Architecture (JTA).

More information on SCPS can be obtain at the following web site: http://www.scps.org/scps. All of the SCPS documentation is available at this site, as well as points of contact and instructions on how to obtain a copy of the reference implementation that was used in this test program.

List of References

1. Muhonen, J., *Space Communications Protocol Standards (SCPS) FY97 DOD Test Plan*, MITRE MTR 97B0000019, April 1997.

2. Muhonen, J., *Space Communications Protocol Standards (SCPS) FY97 Test Procedures*, MITRE MTR 97B0000050, September 1997.

3. The Joint NASA/DOD Space Communications Protocol Standards Technical Working Group, *Recommended Development of Interoperable Data Communications Standards for Dual-Use by US Civil and Military Space Projects*, DOD and NASA Jet Propulsion Laboratory, September 1993.

4. Brakmo, L., S. O'Malley, L. Peterson, *TCP Vegas: New Techniques for Congestion Detection and Avoidance*, Proceedings of ACM SIGCOMM 94, 31 August-2 September 1994, University College London, London, UK.

5. Brakmo, L. and L. Peterson, *TCP Vegas: End to End Congestion Avoidance on a Global Internet*, IEEE Journal on Selected Areas in Communication, Volume 13, Number 8, October 1995, pp 1465-1480.

6. Ahn, J. et al., *Evaluation of TCP Vegas: Emulation and Experiment*, Proceedings of ACM SIGCOMM 95, 28 August - 1 September 1995, Cambridge, MA.

7. Allman, M. et al., *TCP Performance over Satellite Links*, Proceedings of the 5th International Conference on Telecommunication Systems: Modeling and Analysis, 20-23 March 1997, Nashville, TN, pp 456-462.

8. Bruyeron, R., and B. Hemon, *Experimentations with TCP Selective Acknowledgment*, ftp://irl.cs.ucla.edu/papers/sack_exp.ps.gz

9. Fall, K., and S. Floyd, *Simulation-based Comparisons of Tahoe, Reno, and SACK TCP*, Computer Communication Review, Volume 26, Number 3, July 1996, pp 5-21.

10. Mathis, M. and J. Mahdavi, *Forward Acknowledgment: Refining TCP Congestion Control*, Computer Communication Review, Volume 26, Number 4, October 1996, pp 281-291.

11. Jacobson, V., *Congestion Avoidance and Control*, Proceedings of SIGCOMM '88, 16-19 August 1988, Stanford, CA, pp 314-329.

12. Jacobson, V, *Modified TCP Congestion Avoidance Algorithm*, end2end-interest mailing list, 30 April, 1990. Ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail

Appendix A

Detailed Analysis of Selected Transport Test Results

Robert C. Durst

Detailed Analysis of Selected Transport Test Results

In this appendix, we examine the details of the transport test results that are presented in summary in the body of the document. The purpose of this appendix is to examine how TCP and SCPS-TP work a) under nominal conditions, b) under conditions of congestion, and c) under conditions in which data corruption is present. We will examine selected test runs in detail to understand how TCP's mechanisms work in these environments, and to compare TCP's to corresponding mechanisms in the TP.

A.1 Nominal Case Behavior

This section examines the behavior of TCP and TP during a 2,000,000 bps test in which neither congestion nor data corruption are present.

A.1.1 TCP Behavior

TCP behavior has been widely discussed in the literature, and its behavior over satellite links has received some attention [7,8]. In this section, we will discuss TCP's behavior over the ACTS satellite, when operating in the configuration shown in Figure 3 (from the body of this report).

In particular, we wish to examine TCP's slow start behavior, the mechanism that TCP uses to gradually increase its transmission rate while determining the capacity of a communication path. We will examine the effects of buffer sizes and "Ack clocking" on slow start in our examination.

To begin with, slow start is a behavior that is invoked at the beginning of a TCP connection (and at certain other times during the connection). Slow start allows the TCP connection to submit a small amount of data to the network initially, then to steadily increase the amount of data submitted to the network every round trip time. The rate of increase is exponential, theoretically at a rate of $s = 2^N$, where s is the number of TCP segments sent in a round trip time, and N is the number of round trips since the connection was established. In fact, most TCP connections do not increase their transmission rate quite this quickly, for reasons that we will discuss later. The value s, the number of segments sent in a round trip time, times the size of a data segment, in bytes, is known as the *congestion window.* This is a limit on the amount of data that TCP may have outstanding during a round trip, and therefore constrains the instantaneous throughput of a connection.

Slow start is important in a general networking environment because it allows resources (such as buffer memory) to be allocated along the communication path in a controlled manner. Further, it allows TCP to *relatively* quickly determine the capacity of the network at a given point in time. Slow start is a required element of standard TCP, and its behavior has special significance to satellite users. Note, in the equation above, that we discuss slow start in terms of *segments* per *round-trip time*. These units are important to satellite users for two reasons.

First, TCP connections that follow communication paths that include a satellite hop experience round trip times that tend to be higher than those of terrestrial-only paths. This means that TCP connections operating over satellite channels will increase their congestion windows more slowly than those utilizing only terrestrial paths. More importantly, if the two are contending for buffer resources in a router, the more “agile” terrestrial connection will consume those resources more quickly than the satellite connection.

Second, the *segments* portion of the units is important to satellite users. For a given channel capacity and round trip time, a connection using a larger segment (packet) size will require fewer round trip times to accelerate to the channel capacity than a connection using a smaller segment size. However, large segments can be more prone to bit-errors than small segments. Hence, the satellite user may be restricted to smaller segment sizes than the terrestrial user because of the error conditions on the satellite link. (Note, though, that in Figures 4 and 6 that at a bit-error rate of $1x10^{-6}$, the throughput for 1400-byte segments with TCP was significantly higher than that for 512-byte segments.)

With this introduction to TCP slow start in mind, examine Figure A-1. It is a plot of TCP sequence numbers (divided by 1024) that are transmitted as a function of elapsed time. This shows the progress of a TCP data transfer. (Recall that TCP uses byte count as its sequence numbering mechanism. This means that the y-axis of the graph can be viewed as an indication of kilobytes transmitted at a particular point in time.) Note that in the first seven seconds of the run, we see behavior that looks consistent with an exponential increase in transmission rate, as we would expect if the connection were behaving according to the slow start exponential growth equation given above.

However, after seven seconds, the curve becomes quite linear, and remains so for the remainder of the session. Why is this so? This is so because the amount of data that TCP can transmit is limited by more than just the current size of the congestion window. It is also limited by the capacity of the sender’s buffers and by the capacity of the receiver’s buffers. The sender’s buffer space is allocated locally, and is known to TCP. The receiver’s buffer space is called the *advertised window*, because it is “advertised” by the receiver in the window field of the TCP header. It is a measure of the buffer space that the receiver has allocated to store data from the sender, and the sender may not have more data outstanding than is permitted by the receiver’s advertised window. Additionally, the sender may not send data that it cannot store for retransmission, so the sender’s buffer space also places a limit on the amount of data that can be outstanding. In the absence of loss, these two buffers define the upper limit on the amount of data that can be outstanding in one round trip time. A round trip time is used as the standard time quantum because that is the amount of time that is required to send the data to the receiver and for the receiver to acknowledge it. In Figure A-1, the curve becomes linear at approximately seven seconds because the sender’s buffers reach capacity. At this point, new data can only be submitted to the network when old data is acknowledged. The rate of acknowledgment is constrained by the capacity of the network (since the data must be received to be acknowledged), so the transmission rate achieves equilibrium. Note that once the congestion window exceeds the smaller of the send and receive buffer sizes, the transmission rate ceases to increase.

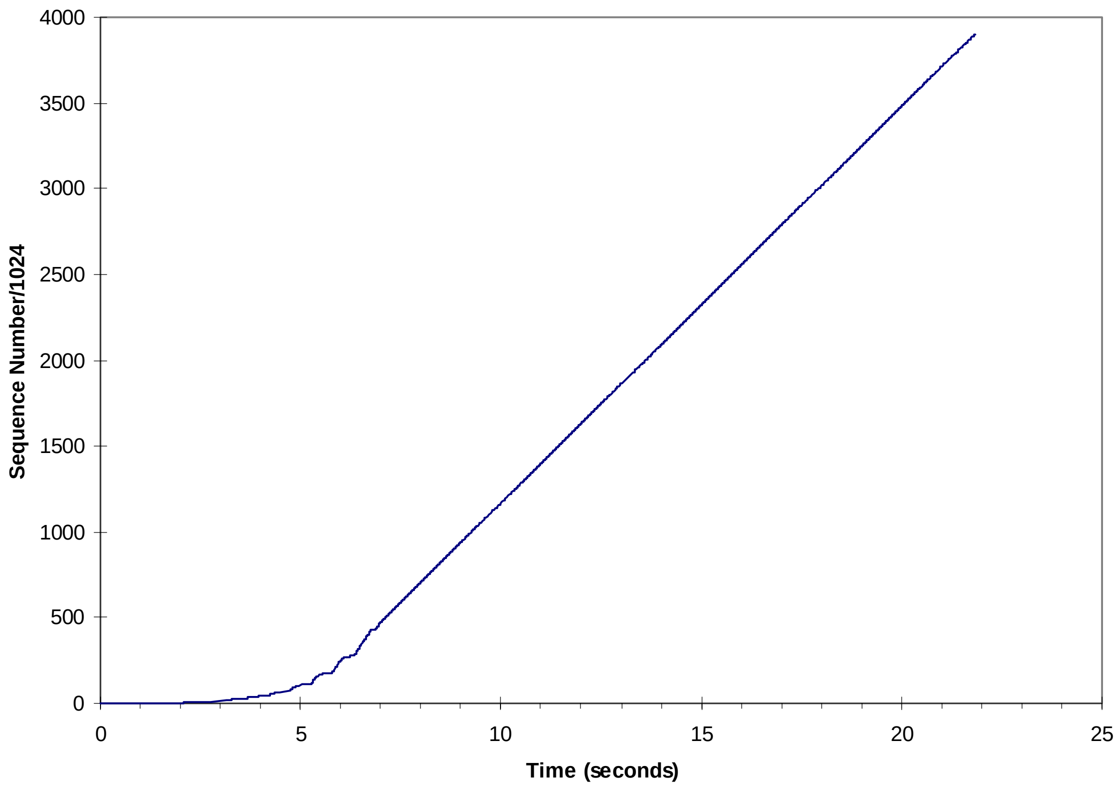


Figure A-1. TCP Sequence Numbers Versus Time Trace

Now consider Figure A-2. It shows a very detailed view of the first five seconds of the TCP connection of Figure A-1. In this graph, both data segments and acknowledgment segments are shown. Data segments are shown as crosses, while acknowledgment segments are shown as dashes. From this examination, we will discuss three things:

1. How the slow start algorithm is put into effect.
2. What "Ack clocking" is, and what it does.
3. What "Delayed acknowledgment" is, and what it does.

The slow start algorithm is quite simple to implement. For every acknowledgment received, the value of the congestion window is incremented by one segment. This allows the segment(s) being acknowledged to be replaced, plus one additional segment to be sent. If every segment is acknowledged, then each segment sent results in two more segments being permitted into the network, resulting in the 2^N growth in the congestion window mentioned earlier.

The practice of using the receipt of acknowledgments (and their effect on the congestion window) to permit transmission of new data is a phenomenon called “Ack clocking”, because the acknowledgments act as a trigger, or a “clock” for the transmission of new data. Look at Figure A-2, at approximately .9 seconds into the run. The cross indicates the transmission of the first data packet of the connection (the cross at time zero is the packet that establishes the connection, and the Ack at .9 completes the connection establishment phase, permitting the initial transmission of data). At time = .9, the congestion window is one segment (its initial value), and the sender transmits a segment, then waits for it to be acknowledged. At time = 1.5, the acknowledgment for that segment arrives at the sender, which causes the sender to increase its congestion window by one segment, to two segments. These two segments are sent immediately, and the acknowledgment for these segments arrives at time = 2.1. Note that a *single* acknowledgment arrives, not one for each segment. This is because most TCP implementations acknowledge every *other* data segment, in order to use less acknowledgment channel capacity. Note that the single acknowledgment means that only three segments, not four, are sent at time = 2.1. At time = 2.6 and time = 2.8, we see another interesting phenomenon: Delayed acknowledgments. When the first two segments that were sent at time 2.1 arrive, the receiver sends an acknowledgment immediately, which arrives at time = 2.6. However, this TCP implementation attempts to acknowledge every *other* segment, so the third segment arriving is not acknowledged immediately. If data were flowing from the receiver to the sender, the acknowledgment would travel on the next outgoing segment (data packets can carry acknowledgment information in TCP). However, no data is flowing from the receiver to the sender. In order to prevent the third segment from time = 2.1 from going unacknowledged indefinitely, TCP’s are required to not delay acknowledgments for more than 0.5 seconds. This implementation of TCP delays acknowledgments for 200 ms, and hence the acknowledgment for the third segment arrives at time = 2.8. Note that the three segments that are sent at time = 2.1 result in five segments being sent between times 2.6 and 2.8.

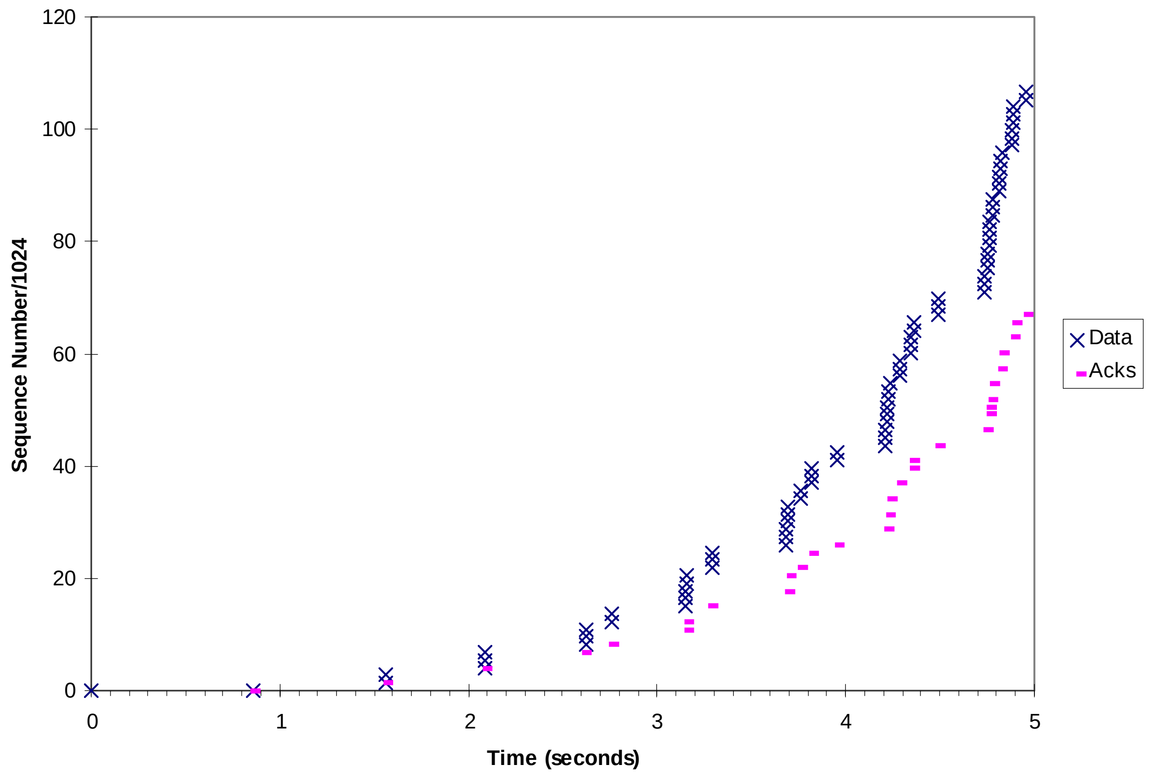


Figure A-2. TCP Slow Start Phase: Data and Acknowledgments

This Ack clocking behavior reduces burstiness of transmissions, as we see in the subsequent round trips, where the transmission of data segments becomes more and more evenly spread across the round trip time. Ack clocking is a remarkably effective mechanism for pacing the transmission of data segments during a connection. However, it depends on enough channel capacity on the return channel to be able to send a steady stream of acknowledgments. If the data rate on the return channel restricts the acknowledgment traffic, the data channel is affected, since data cannot be sent without acknowledgments to “clock” it out. If the TCP implementation is modified to send fewer acknowledgments (fewer than one for every other data segment), the performance of slow start is adversely affected, since the congestion window grows by only one segment for every acknowledgment received. Some commercial satellite service providers have considered discarding queued acknowledgments on low-bandwidth channels (forwarding only the one with the highest acknowledgment number), but this destroys the self-clocking behavior of TCP, resulting in very poor performance.

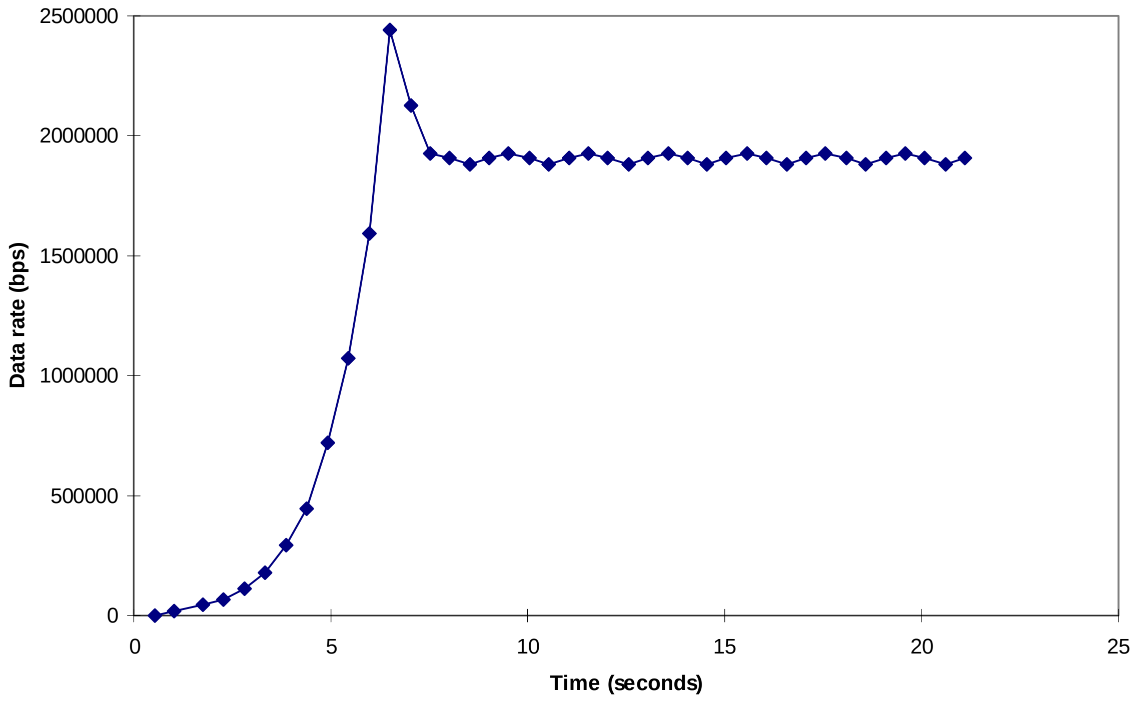


Figure A-3. TCP Transmission Rate versus Time (No errors, No congestion)

Figure A-3 shows the data transmission rate calculated at half-second intervals throughout the duration of the connection. The effect of exponential growth of the congestion window is easily seen in the exponential growth of the transmission rate. Note that the capacity of the channel is 2,000,000 bps, and that at time = 6.5 seconds, the data rate is well over that capacity. What prevents data loss in this case? The combined effect of the sender's limited buffer capacity and large buffers in the router (CISCO 4000 Router #1, in Figure 3 of the body of the document). The router is forwarding packets at 2,000,000 bps. The arrival rate exceeds that value, and a queue builds. If the buffer space at the sender (or receiver) were not limited, subsequent increases in the congestion window (and the rate of transmission) would result in the router's buffers overflowing. However, the sender's limited buffer capacity prevents continued acceleration of the transmission rate, and the steady pacing of data *out* of the router results in a stream of acknowledgments that result in data subsequently being clocked out of the sender at an appropriate rate. At time = 5.5 seconds, the round-trip time (the time between the transmission of a data packet and the receipt of its acknowledgment) was approximately 520 milliseconds. By time = 7.46 seconds, the round trip time had increased to 730 milliseconds. This increase in round trip time is due to queuing delays at the router. The round trip time remains at approximately 710 milliseconds for the remainder of the connection, indicating that the queue at the router persists for the duration of the connection. This, in and of itself, is not particularly significant. However, increased round trip times result in longer time-out values, in the event of a retransmission time-out. More importantly, if a connection consumes router buffer space (approximately 50,000 bytes) for the duration of its connection, that buffer space is not available to absorb the transient bursts of data from other connections. (Readers should note that the data rates shown in this and similar graphs are calculated in terms of *user data only* – no headers are included in the data rate calculation. As a result, the numbers appear to be lower than the maximum capacity of the network. The difference is primarily the portion of network capacity consumed by headers.)

A.1.2 SCPS-TP Performance

The SCPS Transport Protocol is an implementation of the TCP and UDP protocols, and includes several extensions to TCP. Of particular interest in this section are SCPS-TP's adaptation of the TCP-Vegas congestion control mechanism and SCPS-TP's rate control implementation.

The essence of TCP-Vegas's congestion control scheme is that the protocol monitors changes in the throughput once per round trip, and adjusts the size of the congestion window accordingly. TCP Vegas has a version of slow start that behaves in a similar fashion to the congestion control algorithm that standard TCP uses, but with a significant difference. While, in theory, TCP doubles its congestion window every round trip while in slow start, TCP-Vegas doubles its congestion window every *other* round trip. The reason for this is to measure the effect on the network of the traffic at a particular congestion window level before further increasing that level. This is a two-edged sword in a high bandwidth-delay product and high delay environment. Firstly, it theoretically takes TCP-Vegas *twice* as long as TCP to open its congestion window in slow start. (As we saw in the previous section, theory and practice differ - since TCP does not acknowledge every packet, but rather every other packet, its opening of the congestion window in slow start is slower than the theoretical value.) On the other hand, by opening its congestion window more slowly than TCP, TCP-Vegas has the ability to determine whether its offered load is causing queuing in the network (as evidenced by a reduction in throughput).

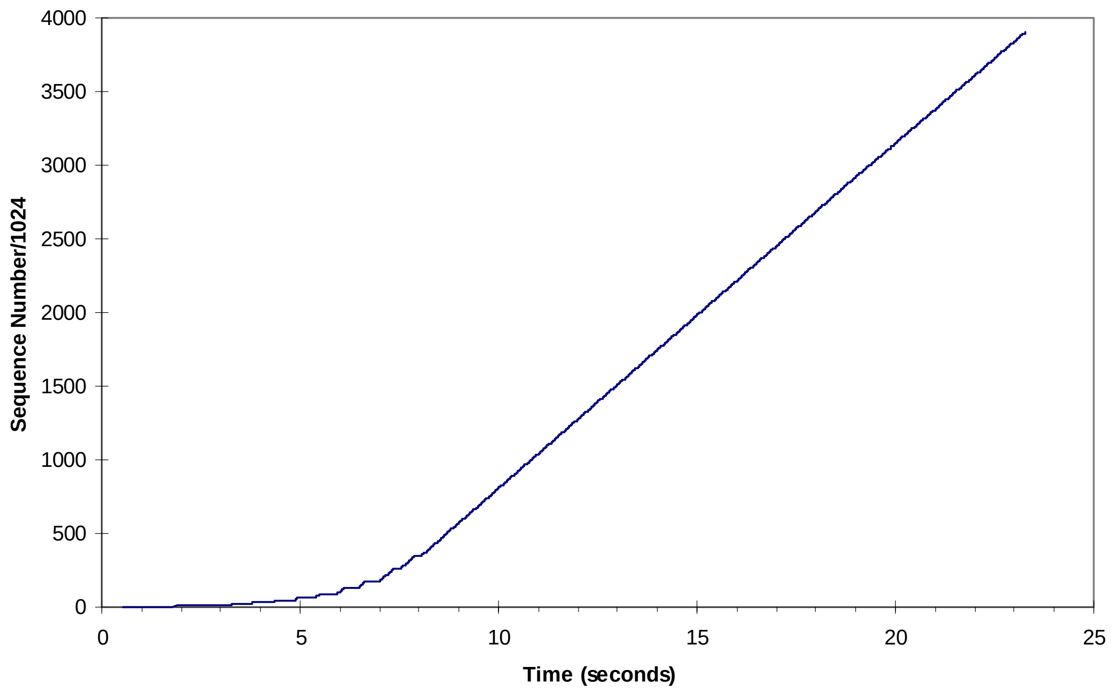


Figure A-4. SCPS-TP Sequence Number Versus Time Trace

TCP-Vegas monitors the round trip times and uses those times to estimate the connection's throughput once per round trip time. Decreases in throughput are interpreted as queuing in the network. Once TCP-Vegas detects a certain level of queuing, it concludes that the network is saturated, and does not increase its offered load. In an environment in which the congestion windows are large and the delays are long, this can prevent buffer overflows and the resulting delays due to retransmission.

When TCP-Vegas detects queuing in the network, it exits slow start and begins its "linear mode" (or congestion avoidance) behavior. In this mode, it samples the throughput once per round trip time, and either increases the congestion window by one segment, leaves it unchanged, or decreases it by one segment. This is in contrast to TCP's behavior of continuing to increase the congestion window until a loss is experienced.

Consider Figure A-4, a sequence number versus time trace for SCPS-TP in comparison to the TCP trace shown in Figure A-1. The appearance of the two traces are generally similar. However, note that while the curve in Figure A-1 "goes linear" at about 7 seconds, the curve for SCPS-TP does not become linear until approximately 8.5 seconds. The difference is that the SCPS-TP's TCP-Vegas congestion control algorithm is growing its congestion window by a factor of two every *other* round trip. The slope of the linear portion of the graph is the same as in Figure A-1, indicating that the "terminal velocity" of the two connections is the same. However, the limiting factor for the SCPS-TP connection is *not* the availability of buffer space. Rather, the TCP-Vegas algorithm will limit the transmission rate, and it is augmented in this case by the SCPS-TP's rate control algorithm, which prevented the connection from exceeding 2,000,000 bps. We will discuss the rate control algorithm later in this section.

Let us examine Figure A-5, which is a detail of the first six seconds of the connection shown in Figure A-4. As before, this graph shows the sequence numbers shown in outgoing data segments and in incoming acknowledgments. Note that at time = .5 seconds, the first data segment is sent, and the acknowledgment for it is received at time = 1.1 seconds. Unlike the TCP trace in Figure A-2, the SCPS-TP's TCP-Vegas algorithm does *not* submit two segments to the network in response to this acknowledgment. Rather, it submits a single segment again and measures the throughput of the connection for the round trip between 1.1 seconds and 1.6 seconds. Since the throughput (as calculated from the offered load and the round trip time) at time 1.6 seconds is undiminished (from the throughput that would have been seen using the offered load and the *best* round trip time to date on the connection), the congestion window is doubled. Two segments are sent at time = 1.6 seconds, and acknowledged at time = 2.2. Two more segments are sent at time = 2.2, and another throughput measurement is taken. Since the throughput is again undiminished, the congestion window is doubled again.

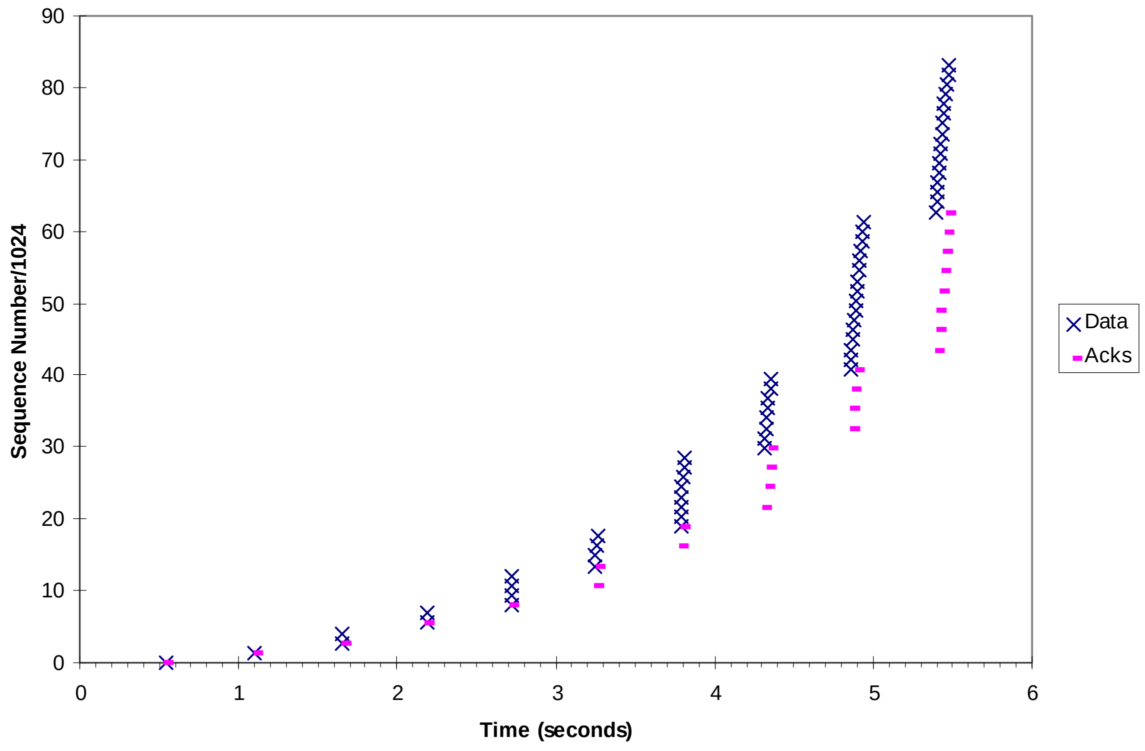


Figure A-5. SCPS-TP Slow-Start Phase: Data and Acknowledgments

This version of slow start continues until a queue builds in the network. At that point, the round trip time will increase, and the throughput measurement will show a decrease in throughput. The threshold for just how much decrease causes an exit of slow start is tunable, but we noticed little sensitivity to it in these tests. This is probably because the SCPS-TP rate control algorithm limited the transmission rate *before* a queue built in the router. In this case, the rate control algorithm had the same effect as the limitation on buffer space did with the TCP connection, but without the formation of a large persistent queue at the router. (The round trip time at the beginning of the connection was .53 seconds, while it was .60 at the end

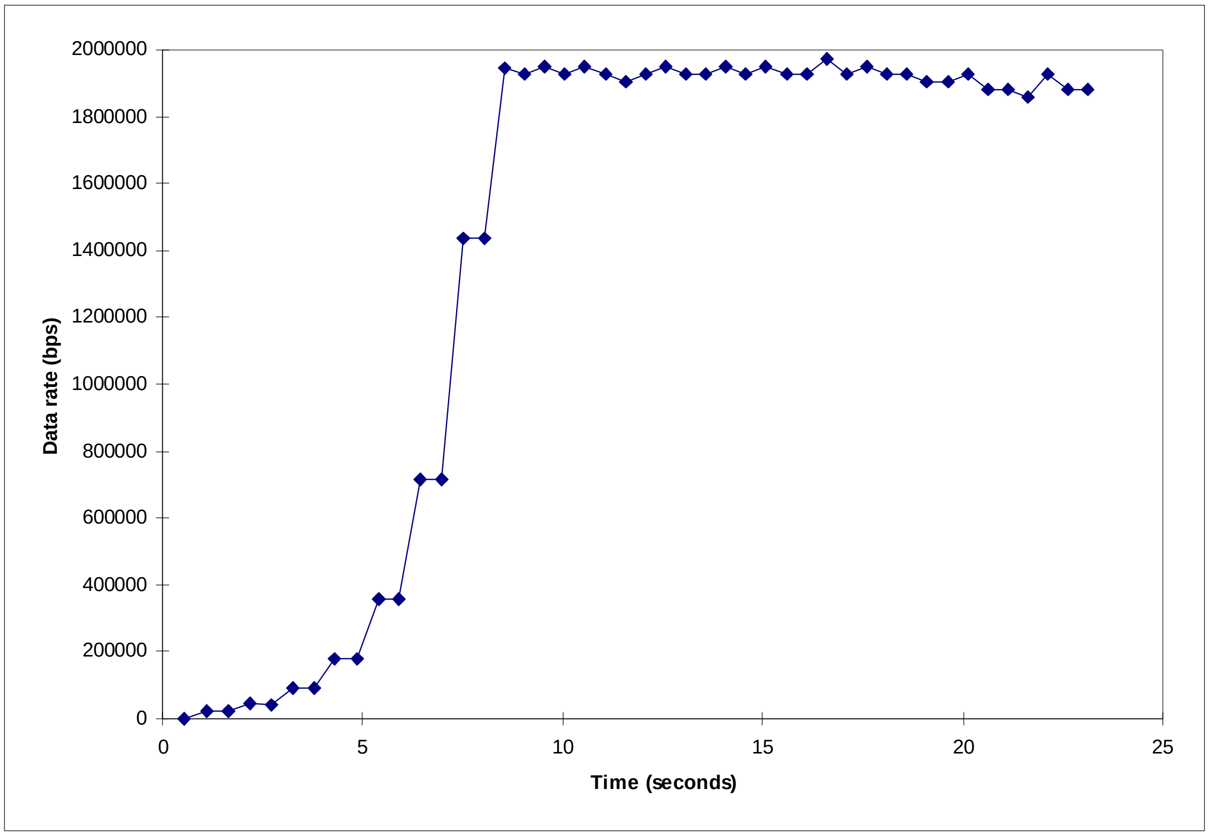


Figure A-6. SCPS-TP Data Rate versus Time (TCP Vegas Congestion Control Enabled)

of the connection, indicating that there was a 70 millisecond, or 12-packet[1], queue at the router. We will see later that the ability to have large buffers at the end systems – on the order of *twice* the bandwidth-delay product of the network, rather than equal to it, are an advantage in the event that the link becomes corrupted (assuming that we do not overrun buffers in intermediate routers as a result of having these larger buffers at the end systems).

---

[1] The 12-segment estimate is derived by knowing that the data rate on the link is 2,000,000 bps and that the packet size (including headers) is approximately 1456 bytes:

$$packets\_in\_queue = \frac{0.070\sec * \dfrac{2000000bps}{8bits\_per\_byte}}{1456bytes\_per\_packet} = 12.0\,packets\,.$$

The effect of the strategy of increasing the congestion window every other round trip is dramatically shown in Figure A-6, which shows the data rate of the connection as a function of time. Once the connection approaches the maximum rate allowed by the rate control parameters, the data rate levels off.

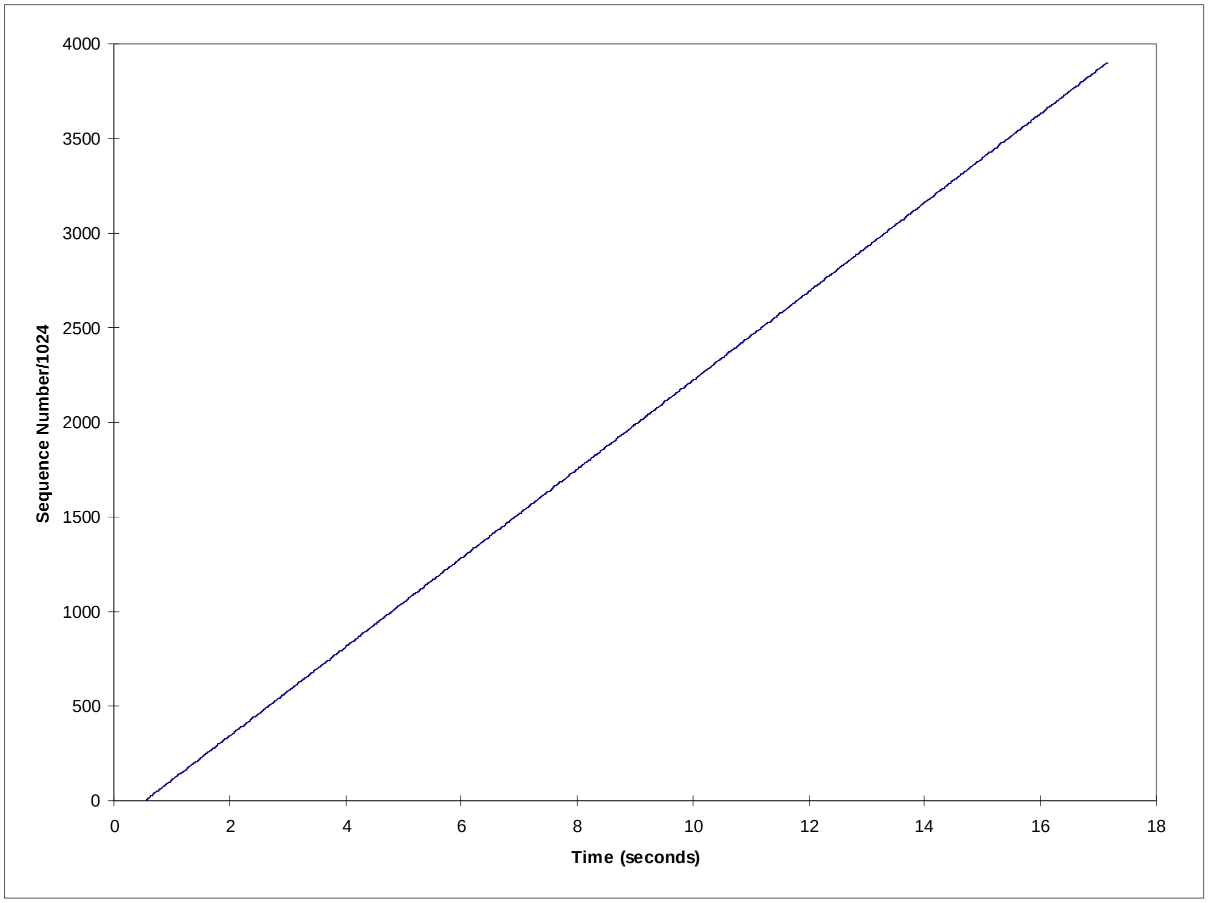


Figure A-7. SCPS-TP Sequence Number Versus Time Trace (Congestion Control Disabled)

As we mentioned previously, the SCPS-TP has a rate control mechanism. The rate control in our implementation is based on a simple "Token-Bucket" algorithm, in which a certain amount of rate control credit (in bytes) is added to the connection's "bucket" once per rate-control interval. In this implementation of the SCPS-TP, the rate-control interval is 10 ms. To effect a 2000000 bps rate control value, 2500 bytes of credit are added to the rate control "bucket" every 10 ms. Before a packet is sent, the protocol ensures that there is sufficient credit in the bucket to “cover” the packet (including headers). (To be precise, our implementation associates the rate control bucket with a *route*, not a particular connection.)

This mechanism can be used independently of the TCP-Vegas congestion control algorithm, either to enforce an apportionment of channel capacity among users, or just to ensure that the protocol does not overrun the available resources. As mentioned in the body of this report, in simple topologies, the use of rate control without congestion control may be desirable. This may also be desirable in reserved-capacity situations, such as may be configured through the use of the Resource ReserVation Protocol (RSVP [RFC 2205]), which is not discussed in this report.

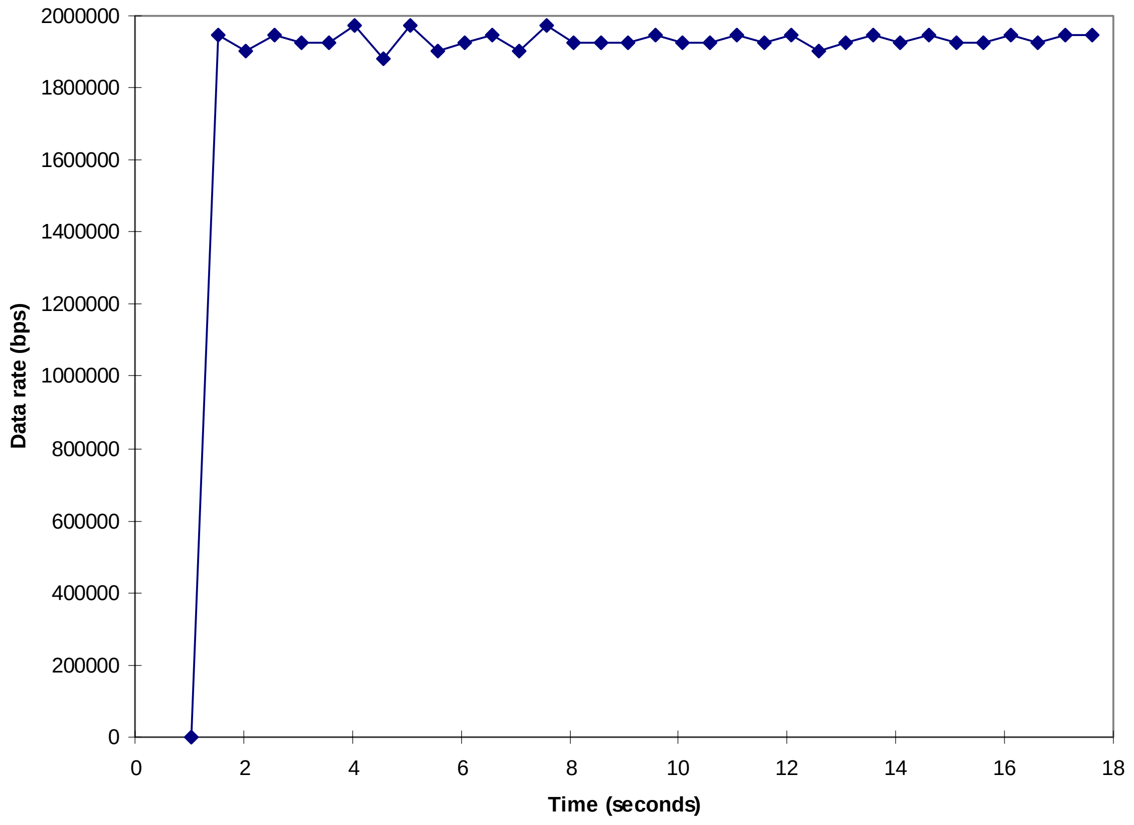


Figure A-8. SCPS-TP Data Rate versus Time (Congestion Control Disabled)

Figure A-7 shows the sequence number versus time trace for a SCPS-TP connection that uses the rate control algorithm, but *not* congestion control. As we can see from the graph, there is no exponential increase in transmission rate. The connection just sends data at the allotted transmission rate as long as there is data to send. This is shown even more clearly in Figure A-8, which shows the data rate of the connection versus time. It should be noted that the use of rate control requires that the SCPS-TP be configured with the correct rate – a setting that is too low will result in under utilization of the communication resources, while a setting that is too high may result in queuing or even congestion collapse. In examining the post-test data for the test shown in Figure A-8, we noticed that round trip times increased over the course of the test, indicating that a queue formed at the router. This queue appears to be approximately the size of that formed by TCP, and considerably larger than that formed by TCP-Vegas.

Although there are potential problems with the use of rate control as the only means of controlling the transmission rate on a connection, these are reasonably well understood, and should be easily accommodated if the communication environment is sufficiently controlled.

The TCP-Vegas congestion control algorithm appears to work well in this environment. However, as already noted, its slow start behavior is more conservative than standard TCP's, resulting in greater delay opening the congestion window. This is an issue that may deserve further consideration – it may be possible to merge the slow-start behavior of standard TCP with the throughput measurement of TCP-Vegas's slow start. This approach is riskier, in that the intervening routers may become seriously congested before the modified TCP-Vegas responds. The improvement in performance may merit the risk, however.

There are other potential problems with TCP-Vegas that are not evidenced in these tests, but merit at least some discussion. TCP-Vegas depends heavily on round-trip timing, and from that it infers things about the state of the resources in the network. TCP-Vegas uses the best round trip time that it has observed on a connection as the benchmark for throughput calculations and comparisons. This leads to an obvious problem in a mobile environment: round trip times change as a result of things *other* than queuing in the network. Specifically, round trip times can change as a result of changing propagation delays. If the propagation delays change only by a moderate amount, say 10% or less, the acceptable buffering tolerances that may be set within TCP-Vegas may be sufficient to accommodate the changes. The effects of this sensitivity do not, in general, affect the SATCOM environment, but are a topic for further study in a high-data-rate mobile environment.

Changes in connectivity that result in changes in link capacity *should* result in modifications in the congestion window, but are probably best accompanied by a return to slow-start behavior. The SCPS Control Message Protocol (part of the SCPS-Network Protocol) provides signaling of such changes for a mobile environment, but as yet the Internet Control Message Protocol does not.

Another potential problem with TCP-Vegas, that *might* affect a SATCOM environment, is the possibility that TCP-Vegas may start up when the network is *already* congested. In this case, the best round trip observed does not indicate optimal performance. What is needed in this case is an explicit signal of congestion – which, again, the SCPS Control Message Protocol provides in the form of a "Source Quench" message. This capability also exists in the Internet Control Message Protocol, but is largely unused because router manufacturers previously did not know how to reasonably control the rate at which these messages are sent. However, recent developments within the Internet community regarding Explicit Congestion Notification should address this issue in the near future.

Let us turn now to a comparison of the three mechanisms discussed so far: standard TCP, SCPS-TP with TCP-Vegas congestion control, and SCPS-TP with rate control. Figure A-9 shows such a comparison. Note that all three approaches result in the same final slope in this sequence number versus time graph, and that the differences are only in how the connection starts up. Note also that this is data taken in the most benign of environments – there was neither congestion nor corruption present, and none of these connections lost a single packet.

In the subsequent sections of this Appendix, we will concentrate on the behavior of TCP and SCPS-TP in less benign conditions. We will consider first the problems resulting from the presence of congestion traffic, and then from the problems of data corruption.

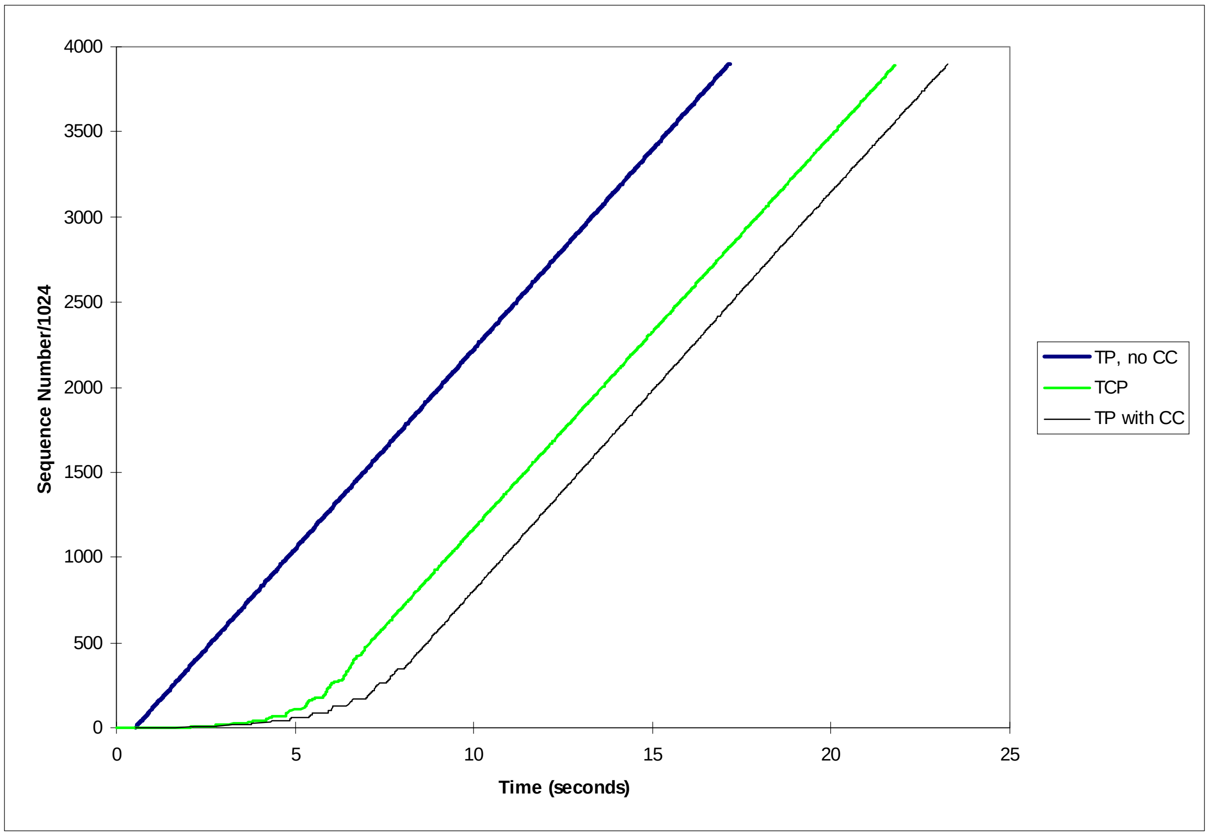


Figure A-9. Comparison of TCP with TP -- With and Without Congestion Control

A.2 Congestion Performance

In 1988, congestion was recognized as a major source of performance degradation in the Internet. [11] The current, widely-distributed mechanisms within TCP for responding to congestion were proposed in 1988, and then revised in 1990. [11, 12] However, in today's SATCOM environment, congestion is considered to be less of a problem than data corruption. As military doctrine evolves toward "coordinated operations without prior planning," the probability of unplanned communication (and therefore congestion) will increase. It is therefore important to consider congestion control and its effect on satellite communication.

A.2.1 TCP Performance

As one would expect from the above discussion, TCP's congestion response has proven to be effective in terrestrial networking. It is also effective in a SATCOM environment, but less so than in a terrestrial environment, because some elements of congestion recovery are a result of round-trip delay. In this section, we will examine TCP's performance in a congested SATCOM environment.

We generated the congestion traffic using an application that randomly selected a congestion traffic generation rate (between 0 and 1,000,000 bps, in these examples), and generated traffic at that rate for five seconds. The application then selected a different random number and set its rate accordingly. The purpose of the test was to examine how TCP's congestion control mechanisms responded to contention for router and link resources.

We will first examine TCP's performance using a data rate versus time plot similar to the one we introduced in Figure A-3. The graph in Figure A-10 differs from Figure A-3, in that it has a trace showing the congestion traffic, as well as the TCP traffic. Readers should note that during the test shown in Figure A-10, only one TCP segment was lost, at approximately t = 23 seconds (we will examine this loss and recovery in Figures A-13 and A-14).

On first blush, we can see from Figure A-10 that TCP is, indeed, responding to congestion in what appears to be an appropriate manner: When the congestion traffic rate goes up, TCP reduces its transmission rate. When the congestion traffic rate goes down, TCP increases its transmission rate. On closer examination, we see some interesting phenomena. First, recall that the capacity of the communication channel is 2,000,000 bps. Any offered load in excess of this will be queued at the router, and if the queue grows bigger than the router's buffer capacity, data will be lost (as at time = 23 seconds).

Notice the TCP traffic data point at time ≅ 5 seconds (data rate ≅ 1,100,000 bps). Prior to this point, the increase in TCP's rate of transmission looks consistent with the graph in Figure A-3. However, between time = 5 and time = 7 seconds, the data rate is still increasing, but not as quickly as before. We get an indication of the answer by examining Figure A-11, which shows the sum of the two curves in Figure A-10. The bold line at 2,000,000 bps indicates the capacity of the satellite link. Values above this line will cause queuing at the router, and if sustained for too long, will cause loss.

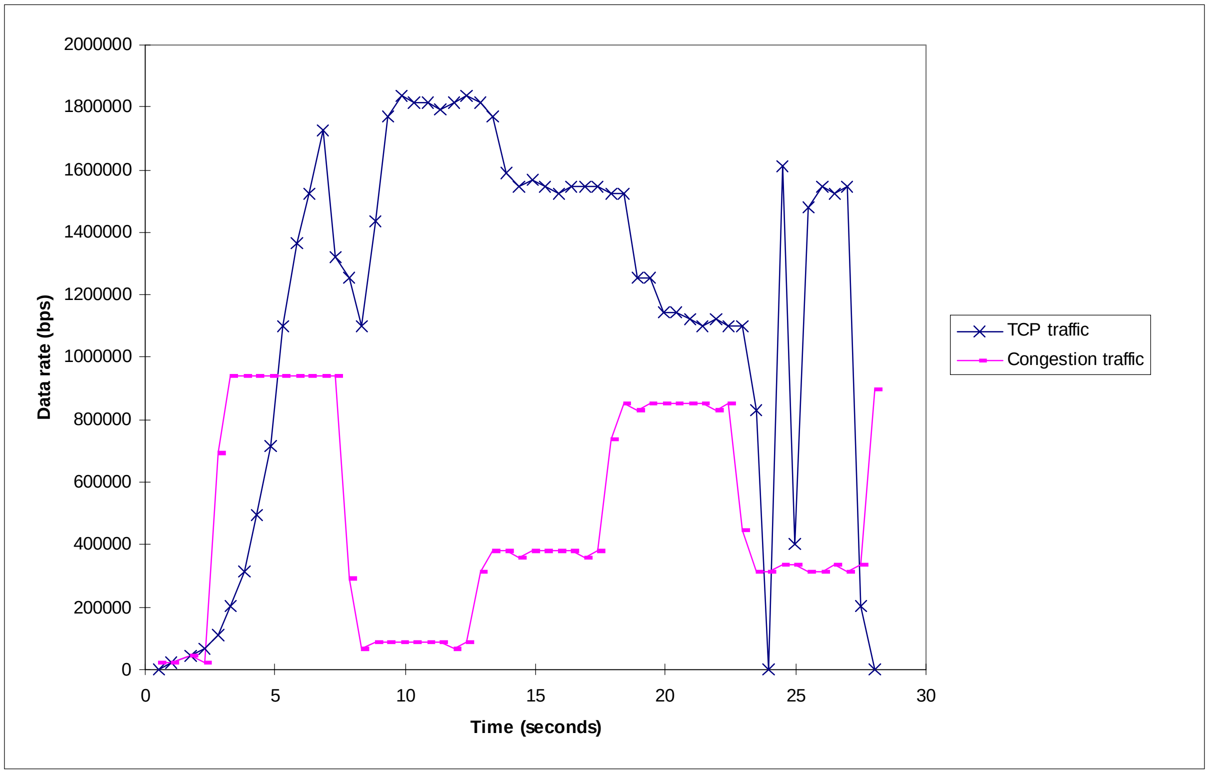


Figure A-10. Data rate versus time: TCP's response to congestion

Between time ≅ 5 seconds and time ≅ 7 seconds, the TCP's rate of transmission still increases, but not as fast as we would expect. The reason for this is that there is other traffic (the congestion traffic) interleaved in the router's buffers, slowing the rate at which *TCP* traffic leaves the router. This slow-down in TCP traffic results in a slow-down in the returned acknowledgments, which, in turn, slows the growth of the congestion window. Why, then, does the TCP rate of transmission drop dramatically between times 7 and 8.5? The answer to this, based on examination of the packet logs, is that the sender's buffers fill at time ≅ 7 sec. At this point, the sender can only replace data that leaves the network, it cannot increase the amount of data in the network (since any data that TCP sends must be buffered for retransmission, and its retransmission buffers are full). As mentioned earlier, the presence of the congestion traffic reduces the rate at which TCP traffic leaves the router, which reduces the rate that acknowledgments are returned, which reduces TCP's ability to introduce new data to the network.

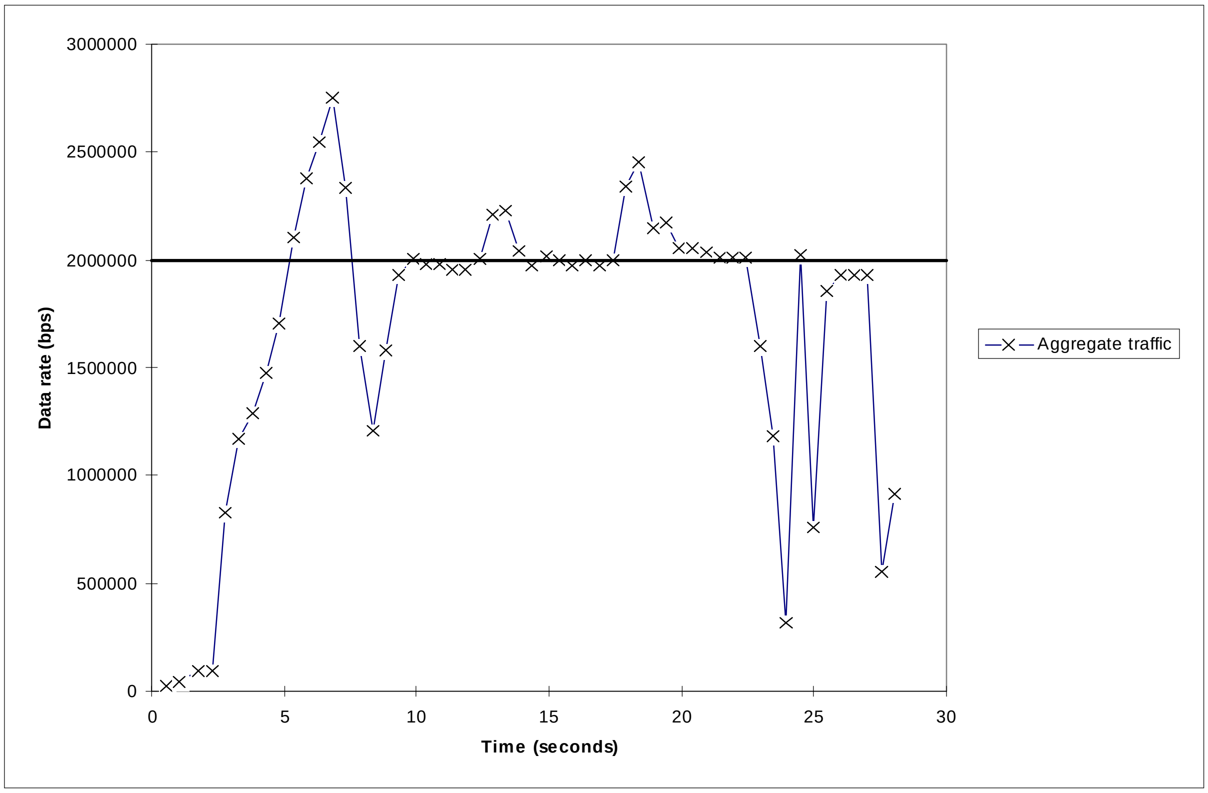


Figure A-11. Aggregate Data Rate for TCP and Congestion Traffic

At time ≅ 8.5 seconds, the congestion traffic drops off precipitously. The effect of this is to allow the TCP traffic to consume the entire link, increasing the rate at which acknowledgments are generated at the receiver. In addition, the round trip times are reduced due to the absence of congestion traffic, so the sender's buffer usage is reduced. This allows the sender to continue its exponential growth of the congestion window, accounting for the rapid increase in transmission rate between times 8.5 and 10 seconds. This acceleration and deceleration of TCP transmission rate continues until time = 24 seconds, when TCP's transmission rate drops to zero. This is because a loss occurred, due (presumably) to a buffer overflow in the router. When we examine Figure A-12, we see a period of time before the loss during which the aggregate traffic exceeds the capacity of the outbound satellite link, but this appears to be much smaller than the episode at time ≅ 5 seconds to time ≅ 8 seconds. Why did the loss occur? Our hypothesis was that a persistent queue had built at the router over the course of the communication.

Using round-trip time information from the packet logs, we attempted to estimate the amount of buffer space in use at the router. We did this by knowing that the router's output rate was 2,000,000 bps, and making the assumption that there were no other significant sources of queuing in the system. This is generally a reasonable assumption in our test environment, since we determined that the destination workstation could consume data at a rate of much greater than 2,000,000 bps. We assumed that the queuing of acknowledgments on the return path was negligible. With this set of assumptions, we estimated the queuing delay in the router by comparing the round trip time resulting from each incoming acknowledgment with the best round trip time seen on the connection (when the path was quiescent). This delay change corresponds to the time required for the router to clock out its queue of data before the packet being timed is transmitted over the satellite link. Using the known rate of the satellite link, we can estimate the router buffer usage. A graph of these estimates is shown in Figure A-12.

Note that the data in Figure A-12 is in good agreement with what we see in Figure A-11. As the aggregate traffic exceeds the capacity of the satellite link, a queue forms at the router. What is not clear from Figure A-11 is that even when the data rate drops between times 7 and 8 seconds, the queue does not completely go to zero. It builds further at time ≅ 13, and again at time ≅ 17, finally resulting in data loss at time ≅ 22. We see a large surge in buffer use again at time = 24 seconds, which we will discuss as we examine the dynamics of TCP in response to loss. The discontinuity in the graph, between time ≅ 22 and ≅ 24 is an artifact of our means of calculating the buffer use – when there is a loss, the round trip times that we calculate as part of our packet log are no longer valid. However, we know from the data that the first point at time ≅ 24 indicates that the queue is essentially empty, a result of the combination of the retransmission and the fall-off of the congestion traffic to levels sustainable by the router without queuing.

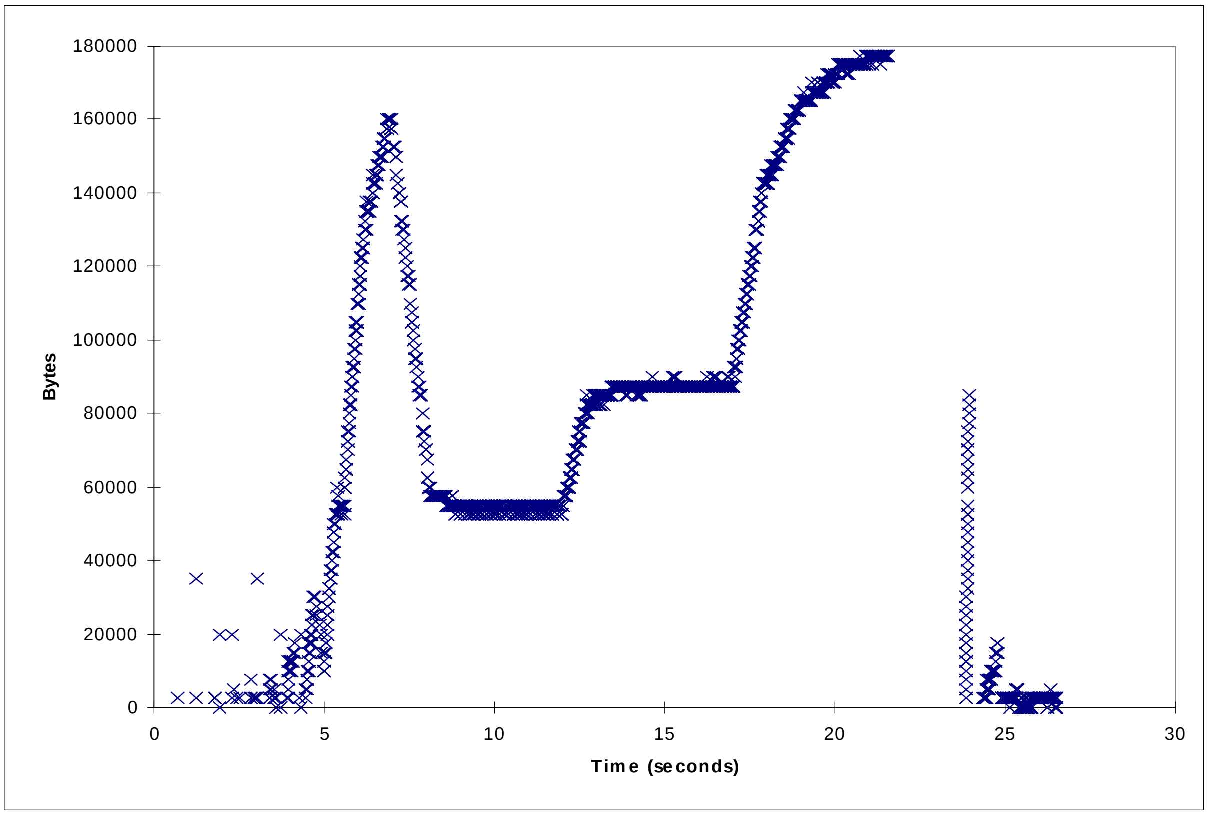


Figure A-12. Estimated Router Buffer Use for TCP Congestion Test

Figure A-13 shows the sequence number versus time trace for the TCP congestion test we have been examining. Note how the changes in slope of this curve at times ≅ 7 and 9 correspond with significant changes in buffer use at the router. Note also that as the queues build in the router, the slope of the sequence number versus time curve flattens (between times 9 and 22). Recall that at this point in the connection, the sender's buffers are full, limiting the connection's data rate. The increasing queuing at the router buffer delays data segments, which in turn delays acknowledgments, which are required to clock out new data segments.

Now consider the curve beyond time 23 seconds. We see a cross that is below the others, followed by a gap, a set of closely-spaced crosses, another gap, and then the transfer continues. The cross that is below the others is a retransmission, due to a loss that appears to have resulted from the router's buffers exceeding their capacity. To this point, the TCP connections we have examined have been executing an algorithm called "slow-start." It is now time to discuss three other algorithms of TCP congestion control: fast retransmit, fast recovery, and congestion avoidance.

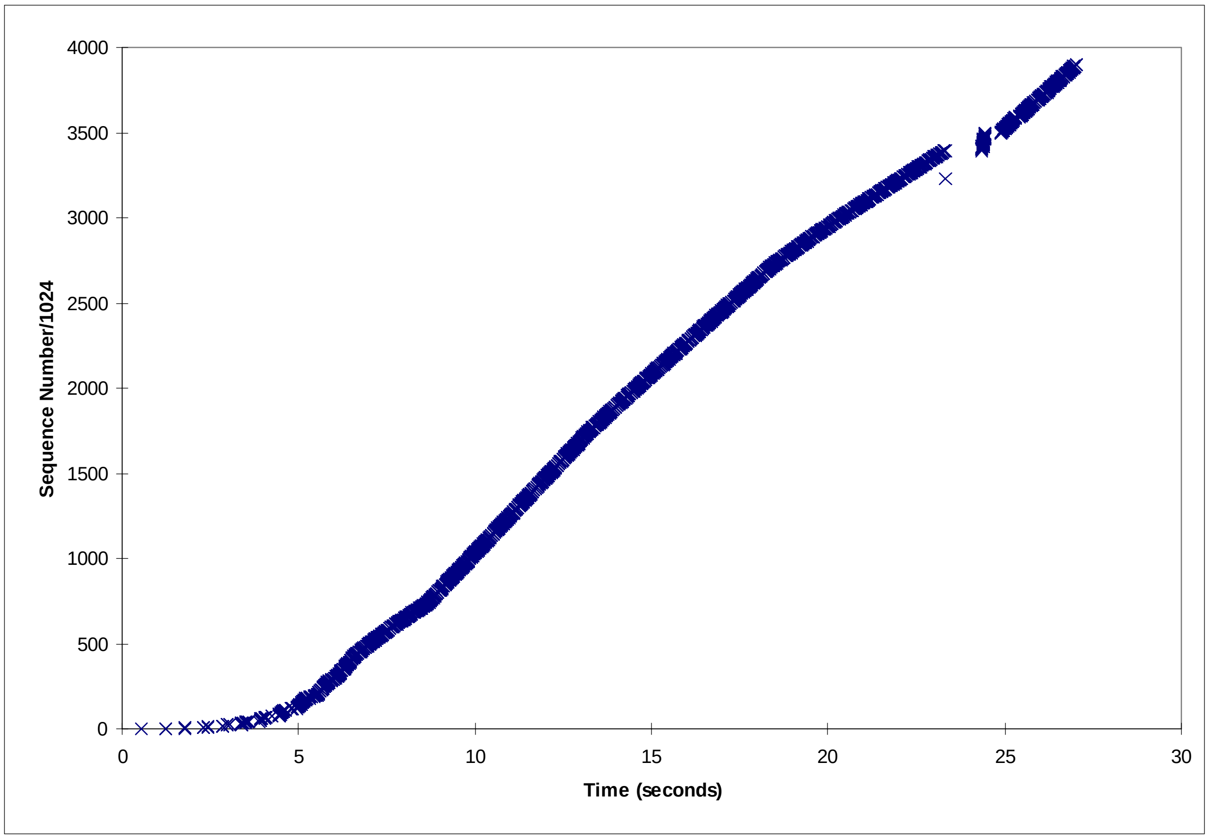


Figure A-13. Sequence Number versus Time Trace for TCP Congestion Test

In Figure A-14, we "zoom in" on the area between times 22 and 26 seconds, to examine how TCP responds to loss. There are a number of factors that will affect this response that we should review briefly. First, satellite propagation delay (round-trip) is approximately 0.511 seconds, and the round-trip time between the source and destination hosts is approximately 0.53 seconds (without any congestion). The data rate of the satellite channel is 2,000,000 bits per second. Therefore, the capacity of the network to carry data "in-flight" (known as the bandwidth-delay product of the network) is given by the following equation:

$$bw \times delay = \frac{2{,}000{,}000 bps}{8 bits / byte} \times 0.53 \sec = 132{,}500 bytes$$

Knowing the capacity of the network is important, since we must configure TCP to be able to submit *at least* this much data to the network in order to fully utilize the network's capacity. Therefore, we use the bandwidth-delay product as a lower bound on retransmission buffer and receive buffer memory sizes.

We can convert this value to a count of TCP segments by dividing by the packet size (including all headers), and we see that the size of the "pipe" (and therefore the minimum expected size of the congestion window) is as follows:

$$bw \times delay(packets) = \frac{132{,}500 bytes}{1456 bytes / \ packet} \cong 91 packets$$

We know that we need to provision the sender and receiver with at least 91 TCP segments of retransmission buffer storage, which at 1400 bytes of user data per segment amounts to 127,400 bytes. For reasons that will become clear later, we do *not* want to exceed twice that amount, so we requested 200,000 bytes of retransmission buffer storage for the sender and for the receiver. For reasons that remain obscure to us, the operating system did not give us the requested 200,000 bytes for send-buffer storage, but rather, gave us approximately 168,000 bytes. This is still greater than the required level, so there was no real problem. We do not know how much space the receiver actually allocated, but it advertised to the sender that it had the amount that we requested. As it turns out, the mismatch between sender buffer space and receiver buffer space led to some "interesting" results that affect TCP's congestion response in ways other than the developers intended.

Recall that by time = 22 seconds, the transmission rate of the connection is limited by ack clocking, since the sender's retransmission buffer is full and new data can only be submitted to the network as old data is acknowledged. The connection is still in slow-start, because no data has been lost. Because the congestion window has been growing by 1400 bytes for every acknowledgment that has been received, it is huge by this point (approximately 1.7 million bytes). The receiver is advertising a window of 200,200 bytes.

The TCP segment that was lost was originally transmitted at time = 22.07 seconds (we determined this by examining the packet logs). While this segment is in transit, acknowledgments continue to arrive. This frees send buffer space, and allows more segments to be transmitted.

The segment transmitted at 22.07 is lost. However, the segments sent after it are not. At time $\cong$ 23.06, the first of these segments arrives at the receiver. Since the segment from 22.07 has been lost, the receiving TCP perceives all segments transmitted after it to be out of order. It saves these on an out-of-sequence queue for later use, and invokes its portion of the *fast retransmit* algorithm. Rather than attempting to paraphrase the description of the fast retransmit algorithm, here is its specification, from RFC 2001:

"3. Fast Retransmit

Modifications to the congestion avoidance algorithm were proposed in 1990 [3] [in this document, reference 12]. Before describing the change, realize that TCP may generate an immediate acknowledgment (a duplicate ACK) when an out-of-order segment is received (Section 4.2.2.21 of [RFC 1122], with a note that one reason for doing so was for the experimental fast-retransmit algorithm). This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order, and to tell it what sequence number is expected.

Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. TCP then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire."

In examining Figure A-14, we see the fast retransmit occur at time 23.34, after the receipt of (what we can determine from examining the packet logs) three duplicate acknowledgments. (In fact, there are four acknowledgments received with the same acknowledgment number before the segment is retransmitted, because the first one is not a *duplicate*.)

Since the receiving TCP is sending a duplicate acknowledgment for each segment that is received out-of-order, the string of duplicates received by the sending TCP continues on well past time = 24. These duplicate acknowledgments perform a critical function in TCP's *fast recovery* algorithm. Here are two more excerpts from RFC 2001 that explain slow start and congestion avoidance, followed by a discussion of the implementation of the congestion avoidance, the fast retransmit and fast recovery algorithms:

Congestion avoidance and slow start are independent algorithms with different objectives. But when congestion occurs TCP must slow down its transmission rate of packets into the network, and then invoke slow start to get things going again. In practice they are implemented together.

Congestion avoidance and slow start require that two variables be maintained for each connection: a congestion window, cwnd, and a slow start threshold size, ssthresh. The combined algorithm operates as follows:

1. Initialization for a given connection sets cwnd to one segment and ssthresh to 65535 bytes.

2. The TCP output routine never sends more than the minimum of cwnd and the receiver's advertised window.

3. When congestion occurs (indicated by a timeout or the reception of duplicate ACKs), one-half of the current window size (the minimum of cwnd and the receiver's advertised window, but at least two segments) is saved in ssthresh. Additionally, if the congestion is indicated by a timeout, cwnd is set to one segment (i.e., slow start).

4. When new data is acknowledged by the other end, increase cwnd, but the way it increases depends on whether TCP is performing slow start or congestion avoidance.

If cwnd is less than or equal to ssthresh, TCP is in slow start; otherwise TCP is performing congestion avoidance. Slow start continues until TCP is halfway to where it was when congestion occurred (since it recorded half of the window size that caused the problem in step 2), and then congestion avoidance takes over.

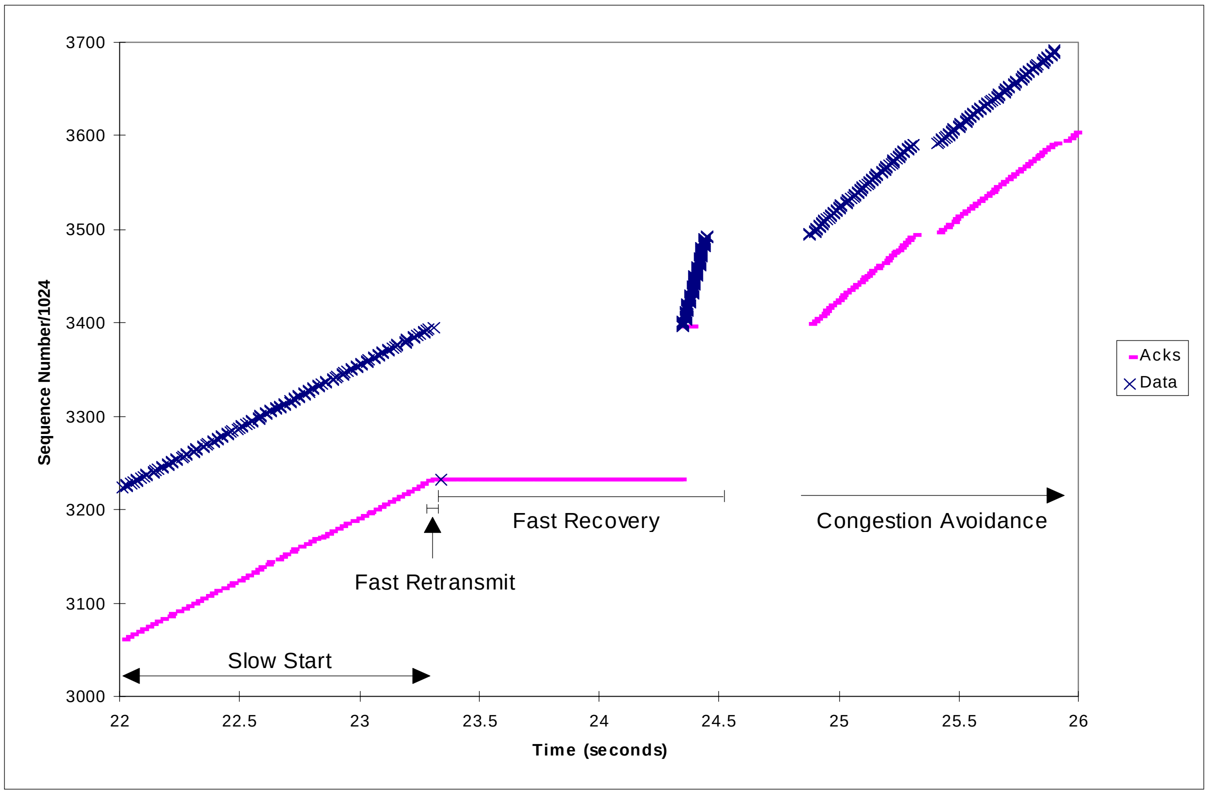


Figure A-14. Detail of TCP's Recovery From Loss

Slow start has cwnd begin at one segment, and be incremented by one segment every time an ACK is received. As mentioned earlier, this opens the window exponentially: send one segment, then two, then four, and so on. Congestion avoidance dictates that cwnd be incremented by segsize*segsize/cwnd each time an ACK is received, where segsize is the segment size and cwnd is maintained in bytes. This is a linear growth of cwnd, compared to slow start's exponential growth. The increase in cwnd should be at most one segment each round-trip time (regardless how many ACKs are received in that RTT), whereas slow start increments cwnd by the number of ACKs received in a round-trip time.

The key things to remember from this for the purposes of our discussion are: In slow-start, the congestion window grows by one segment per acknowledgment received. After a loss signaled by duplicate acknowledgments, the congestion window is set to half its previous value and the connection enters *congestion avoidance*. In congestion avoidance, the congestion window grows by at most one segment per round trip time, rather than one segment per acknowledgment.

Now for the implementation excerpt:

4. Fast Recovery

After fast retransmit sends what appears to be the missing segment, congestion avoidance, but not slow start is performed. This is the fast recovery algorithm. It is an improvement that allows high throughput under moderate congestion, especially for large windows.

The reason for not performing slow start in this case is that the receipt of the duplicate ACKs tells TCP more than just a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and TCP does not want to reduce the flow abruptly by going into slow start.

The fast retransmit and fast recovery algorithms are usually implemented together as follows.

1. When the third duplicate ACK in a row is received, set ssthresh to one-half the current congestion window, cwnd, but no less than two segments. Retransmit the missing segment. Set cwnd to ssthresh plus 3 times the segment size. This inflates the congestion window by the number of segments that have left the network and which the other end has cached (3).

2. Each time another duplicate ACK arrives, increment cwnd by the segment size. This inflates the congestion window for the additional segment that has left the network. Transmit a packet, if allowed by the new value of cwnd.

3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This ACK should be the acknowledgment of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, since TCP is down to one-half the rate it was at when the packet was lost.

The second paragraph provides insight into the use of duplicate acknowledgments in fast recovery – to maintain data flow between the end systems in the event of a loss. Further, the final sentence of this excerpt states the goal of the congestion avoidance phase: to reduce TCP's data rate to one half the level it was when the loss occurred. Van Jacobson, principal author of these algorithms, provides a stronger statement of these goals in his original elaboration of the algorithms in 1990:

"… But, as long as the receiver's offered window is large enough (it needs to be at most twice the bandwidth-delay product), we continue sending packets (at exactly half the rate we were sending before the loss) even after the loss is detected so the pipe stays full at exactly the level we want and a slow-start isn't necessary." [12]

Now, let us examine Figure A-14 and see why neither of these goals is met in this case. (That is, the pipe does *not* stay full, and the transmission rate after the loss is significantly greater than half of the rate at the time of the loss.)

The root cause in both cases is the fact that the sender's buffer is smaller than the receiver's buffer. (This *should not* cause problems, because the sender and receiver are not expected to coordinate buffer sizes – the algorithms should accommodate whatever buffer space is available, as long as the receiver properly advertises its buffer availability, and the receiver's buffer space is between one and two times the bandwidth-delay product of the path. This last requirement can be *very* tricky to determine before the connection is established.)

First let us consider the issue of the pipe not staying full. We see from the graph that the retransmission is sent at time = 23.34 (the exact time comes from the packet logs), and the acknowledgment arrives at time = 24.35. In between these times, a steady stream of duplicate acknowledgments arrives. This tells us that over the one full second (*two* round trip times) that elapses between the retransmission and the acknowledgment of that retransmission, that queued TCP data is steadily arriving at the receiver. This is not particularly surprising, since we see from Figure A-10 that both TCP traffic and the congestion traffic are (and have been) at relatively high rates for some time. Therefore, it is not unreasonable that the TCP traffic in the router's buffers is mixed relatively evenly with the congestion traffic. So the data that one would expect to take just over a half-second to transmit actually takes over a second due to time spent queued at the router. When the acknowledgment for the retransmission does arrive, it acknowledges 168,000 bytes of data – exactly the sender's capacity. *This* explains why the sender went idle for a full second – the sender's buffers were full. As a result, an important aspect of fast recover, that is, the ability to keep the pipe full, is lost.

Before we examine the second issue – the transmission rate after the loss, let us examine the burst of data that occurs just after the acknowledgment of the retransmission arrives. Examining this transmission in detail, we see that the sender emits 72 segments in approximately 100 milliseconds (an instantaneous transmission rate of 8.4 million bits per second). This immediate injection of 72 segments into the network is a cause for concern, because after an idle period of a full second, the router's buffers could be queued from other (terrestrial) sources, and the burst of data could easily cause loss. This did not occur in these tests because the routers were otherwise idle. However, in a more realistic environment, this is a concern.

Now let us examine the data rates that the connection sustained before and after the loss. Based on the intent of the congestion avoidance algorithm, we would expect the data rate after the loss to be half the data rate before the loss. It is not possible to directly measure the data rates (and get a meaningful answer), because the congestion traffic conditions have changed between the time of the retransmission and the time that data starts flowing again. Rather, let us look at the congestion window, and see what the data rates *would* be, all other things being equal. Before the loss, the connection's receive window was 200,200 bytes, the send buffer space was 168,000 bytes, and the congestion window was enormous, since it had been incrementing 1400 bytes for every acknowledgment received since the beginning of the connection. So the constraint on the sender was its own buffer size: 168,000 bytes. The sender can, therefore, send up to 168,000 bytes per round-trip. After the loss, the receive window is still 200,200 bytes, the send buffer space is still 168,000 bytes, but the congestion window is reduced to one-half of the minimum of the previous congestion window value and the receive window size. The minimum of these two numbers is the receive window (200,200), so the congestion window is set to 100,100 bytes. The sender is now authorized to send 100,100 bytes per round trip, rather than 168,000 bytes. This difference is 60% of the previous value, not 50%. The fact that there *is* a discrepancy is more important than the size of the discrepancy. When attempting to adjust the transmission rate of the sender, no account was made of the fact that the sender's buffer size might be the limiting factor to throughput. In the case of a more significant mismatch (where the sender's buffer size is less than half of the receiver's), the adjustment of the congestion window would have had no effect on transmission rate at all. In this case, the router would continue to become congested, lose data, and *eventually* the congestion window would come down to a point where it provided an actual constraint on the transmission rate. However, it appears that a simple fix to this would be to halve the minimum of the congestion window, the receiver's advertised window, *and the send buffer size* when implementing the algorithm.

One final note about the TCP congestion control algorithm. As indicated in the above quote by Van Jacobson, the TCP receive buffer needs to be at least as large as the bandwidth-delay product, *and no more than twice* the bandwidth-delay product of the connection's path. This is because TCP's congestion control algorithm grows without bound – if there is no loss, the congestion window never stops growing. After a loss, the congestion control algorithm cuts the congestion window in half and immediately begins growing it at a rate (not to exceed) one segment per round-trip time. Current design practice in the Internet indicates that a "properly provisioned" router should have at least one bandwidth-delay product's worth of buffer space available[2]. Since the TCP congestion control algorithm can do nothing but grow until there is loss, it depends upon the receive window to limit transmission. This is unfortunate, since the receiver typically knows nothing about the bandwidth-delay product of the network. To properly set the receive window requires manual tuning, which may not be possible due to the application, and is difficult to do in a general-purpose manner.

A.2.2 SCPS-TP Performance

Now let us consider SCPS-TP's performance in a congested environment. For the purposes of comparison, we ran the congestion traffic generator with the same random number seed as with the TCP test discussed in the previous section. The SCPS-TP congestion control algorithm (called TCP-Vegas) does not depend on a maximum size of the send or receive buffers (as does TCP's standard congestion control algorithm). Knowing this, we configured the SCPS-TP transmit and receive buffers to 400,000 bytes. We configured SCPS-TP's rate control parameter to a value just under 2,000,000 bps. (We set the rate control to a value less than the channel data rate, assuming that there was some framing overhead of which we were unaware and for which we had not previously accounted.)

We implement portions of the TCP-Vegas algorithms described in [4, 5, 6]. In particular, we implement the TCP-Vegas version of slow start from [4, 6], not the Vegas* algorithm from [5] (although we may experiment with this at some point in the future). We also implement the TCP-Vegas congestion avoidance algorithm. We do *not* implement the TCP-Vegas Fast Retransmit algorithms. Further, the SCPS-TP has been built with a route-specific

---

[2] There is a problem in calculating this value for a router that has only terrestrial connections into and out of the router, when the *path* that a TCP connection takes passes through the router *and* a remote satellite link. The router will almost certainly be sized for terrestrial bandwidth-delay products, although the router at the satellite hop will probably be appropriately provisioned. However, if the first router is the bottleneck, queuing will occur at *it* rather at the satellite terminus. Its buffers will probably be undersized, in this case.

parameter that indicates whether the protocol should interpret losses as indications of congestion or of corruption. When set to assume that loss is an indication of congestion, the protocol will halve its congestion window in response to loss. When set to assume that loss is an indication of corruption, the protocol makes no modifications to the congestion window, depending on the TCP-Vegas algorithms to make adjustments.

In section A.1.2, we described the TCP-Vegas slow start and congestion avoidance algorithms. To briefly recap, the TCP-Vegas slow start algorithm doubles the congestion window every other round trip time until it detects queuing in the network. At that point it switches to its linear mode, in which it may adjust the congestion window up or down one segment size per round trip (or may leave it unchanged). The decision to adjust the congestion window is, again, based on the protocol's perception of queuing in the network. This perception is formed by monitoring the round trip times, and TCP-Vegas uses those times to estimate the connection's throughput once per round trip time. Decreases in throughput are taken as indications of queuing.

We wished to examine the response of SCPS-TP's TCP-Vegas implementation to the same congestion traffic with which we tested TCP. Figure A-15 shows data rate versus time for a SCPS-TP connection. The "double-the-congestion-window-every-other-round-trip" behavior is especially clear in the range between 0 and 6 seconds. The four data points following that are also slow start – we believe that the non-monotonic nature of these data rate values are a side-effect of the binning algorithm used to generate the data-rate plot. (There are two distinct pairs of values, and we believe that some of the rate that should be attributed to the second point of each pair is being attributed to the first point of the pair. The "correct" value should be approximately half way between the points of each pair.)

It should be noted that there was no data loss during this test.

Now examine Figure A-16, which shows the sum of the two curves in Figure A-15, and compare this to Figure A-11 (the corresponding graph for TCP) between times 7 and 10 seconds. We do not see TCP-Vegas exceeding the maximum output rate of the router by as much as we do in Figure A-11. While it would be nice to attribute this to some particular virtue of TCP-Vegas, in fact, it appears to be due to the fact that TCP-Vegas's slow-start mechanism is enough slower than TCP's that the congestion traffic had moved on to another (lower) rate by the time that TCP-Vegas was peaking its slow start traffic. However, the fact remains that TCP-Vegas's slow start, augmented by the rate control algorithm that we have implemented, did not build a large, sustained queue at the router during slow-start (refer to Figure A-17). It is possible that TCP-Vegas will overrun the capacity of a link and router in slow start. Whenever a congestion window is large and is doubled, this is a possibility, and the authors of TCP-Vegas freely admit this. We use the rate control mechanism to help avoid such overruns, when the capacity of the bottleneck resource is known and there is no competition for the resource.

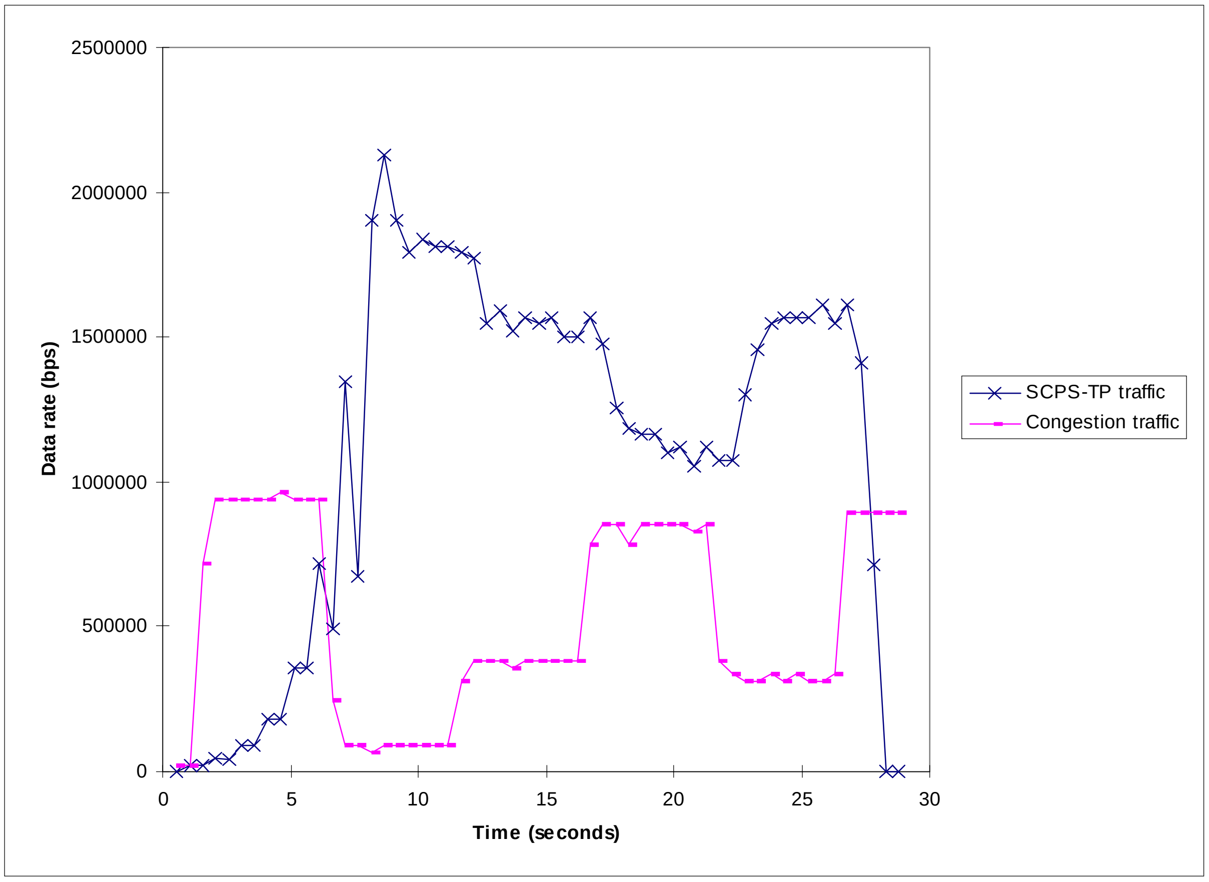


Figure A-15. Response of SCPS-TP Traffic to Congestion Traffic

Note that the downward trends in the SCPS-TP data rates shown in Figure A-15 follow the times when the data rates exceed the capacity of the link (as shown in Figure A-16). The congestion control algorithm is performing as it should. Note also, in Figure A-17, that the buffer use during those times has a downward trend. This is very important – the congestion control algorithm is acting to drain off the queues at the router. If one examines the "plateau" areas of Figure A-12 (TCP's router buffer use), one sees that between approximately 8 and 12 seconds and between 13 and 18 seconds that the buffer use is flat, with occasional upward spikes. The corresponding regions in Figure A-17 show distinct downward trends, indicating that the protocol is acting to reduce buffer use in the router, and therefore reduce the probability of congestion and congestion-caused loss. The important thing about this congestion avoidance algorithm is that *the network did not have to be driven to loss to determine that congestion was present*. Stated differently, packet loss is not the sole indication of congestion. This leaves us free to interpret loss differently – for example, as an indication of a *corrupted* link, rather than a congested path.

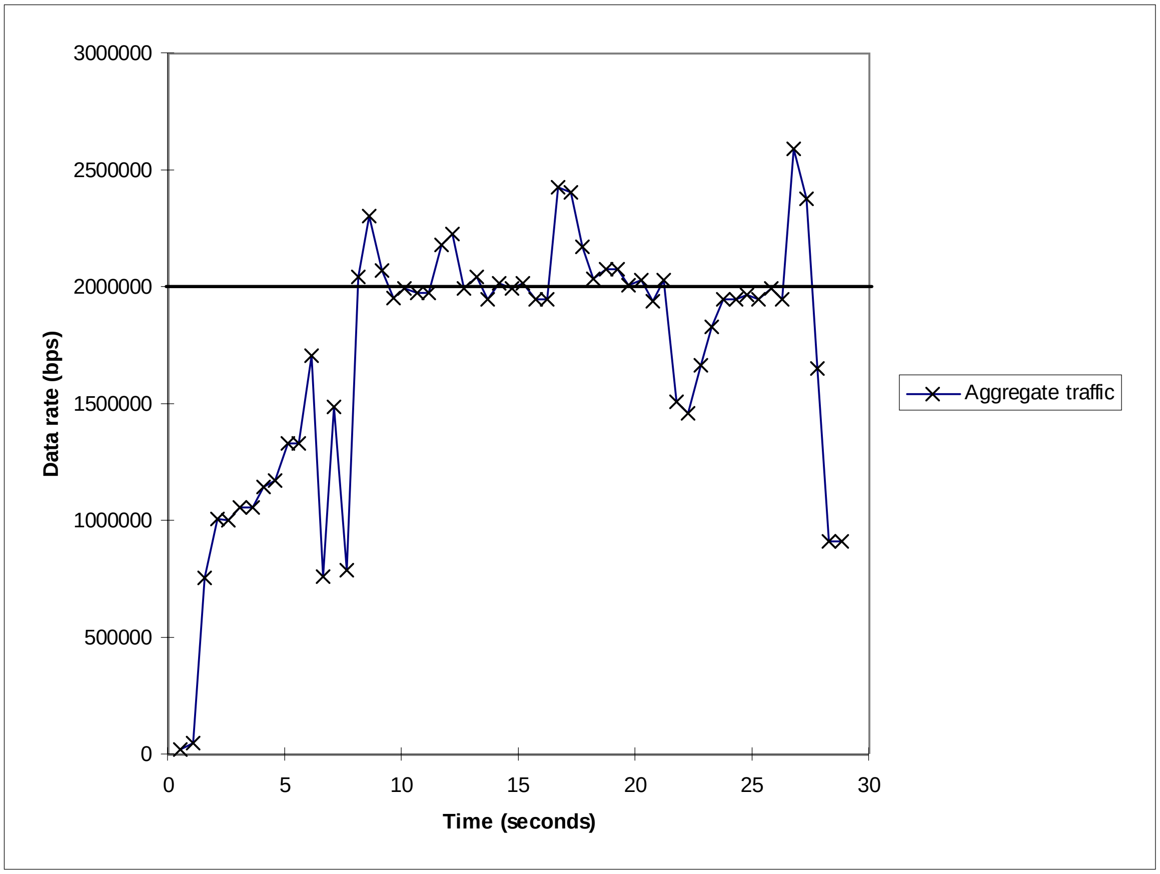


Figure A-16. Aggregate Data Rate for SCPS-TP and Congestion Traffic

In our examination of Figure A-17, and considering how this information would relate to other situations, we have come to the conclusion that the ability to adjust the congestion window (downward) at a rate of only one segment per round trip will not scale well to higher data rate (that this 2,000,000 bps), long-delay paths. When the congestion window is very large, on the order of ten times what it is in these tests, we need the ability to reduce the congestion window much more quickly. We are considering modifications to the TCP-Vegas congestion avoidance algorithms to allow this.

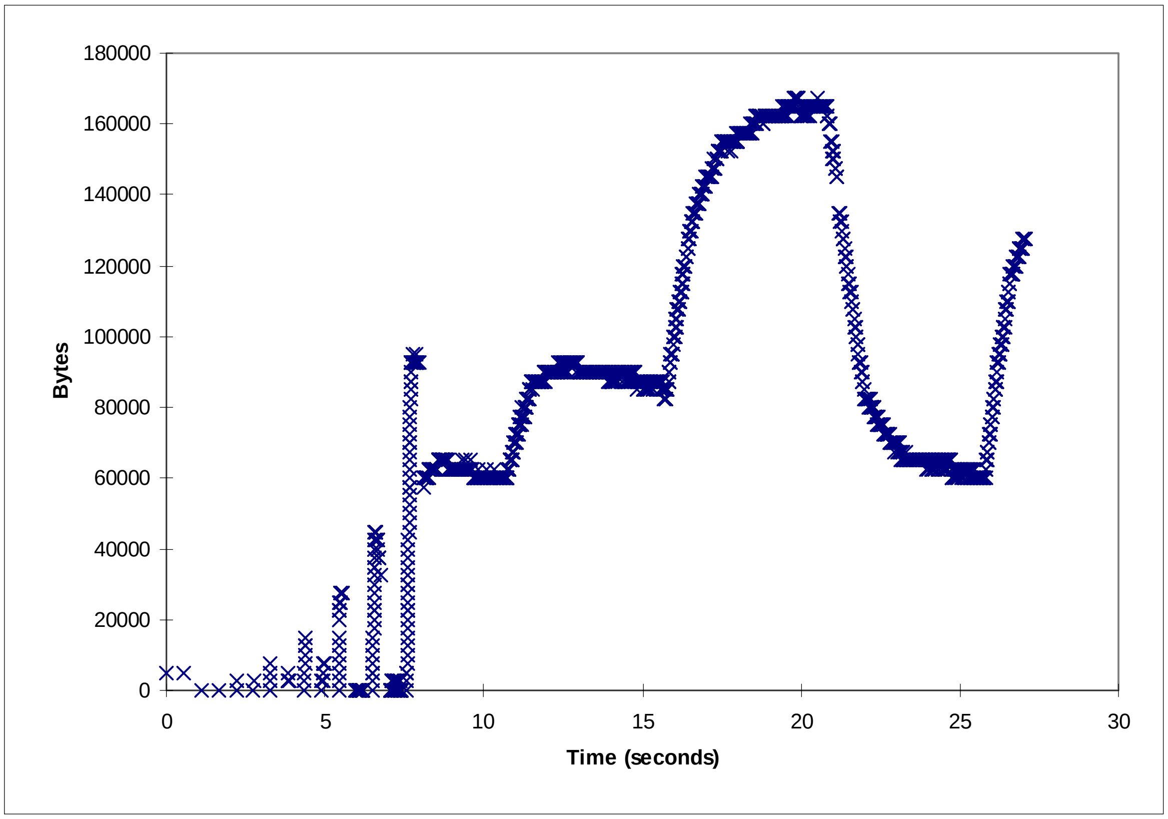


Figure A-17. Estimated Router Buffer Use for SCPS-TP Congestion Test

While TCP-Vegas suffers from notable problems, as mentioned before, we are generally pleased with its performance in these tests. Its overall congestion performance, as discussed in the body of this report, is consistent with that of TCP. As we have seen in our detailed examination, it has some attractive qualities, such as the ability to drain off queues in routers and the ability to detect congestion from indications other than packet loss.

A.3 Corruption Performance

We now examine the response of these protocols to corruption. As discussed in the body of the report, we tested the protocols over a wide range of error rates. In this Appendix, we examine the response to selected error rates in more detail.

To preface this section, discussing the effects of errors on TCP and SCPS-TP performance, we include the following excerpt from Section 1.1 of the TCP specification [RFC 793]:

"Computer communication systems are playing an increasingly important role in military, government, and civilian environments. This document focuses its attention primarily on military computer communication requirements, especially robustness in the presence of communication unreliability and availability in the presence of congestion, but many of these problems are found in the civilian and government sector as well.

As strategic and tactical computer communication networks are developed and deployed, it is essential to provide means of interconnecting them and to provide standard interprocess communication protocols which can support a broad range of applications. In anticipation of the need for such standards, the Deputy Undersecretary of Defense for Research and Engineering has declared the Transmission Control Protocol (TCP) described herein to be a basis for DoD-wide inter-process communication protocol standardization.

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks."

As the use of TCP expanded throughout the Internet, congestion, rather than corruption became a significant problem. In 1986, the Internet experienced the first of several "congestion collapses" [11], which resulted in a desperate need to improve the congestion control abilities of TCP. The simple, elegant solutions that we discussed in the section A.2.1 have served the Internet well since their introduction in 1988 and revision in 1990. However, the solution to the Internet's congestion problems came at the expense of TCP's ability to handle other forms of loss efficiently. Note that, strictly speaking, TCP is still "robust in the presence of communication unreliability." However, in military and tactical environments, the degradation in performance can be severe, as we will see in this section. One of the key goals of SCPS-TP is to retain the ability to respond appropriately to congestion, and to *restore* the ability to appropriately respond to corruption.

A.3.1 TCP Performance

As mentioned in Section A.2.1, TCP uses data loss as an indication of congestion within the network. Since most underlying protocols discard damaged packets, most corruption loss will be interpreted by TCP as network congestion. As we discussed at length in Section A.2.1, TCP's response to congestion is to attempt to halve its transmission rate. Clearly, this is an inappropriate response to corruption loss.

Figure A-18 shows a sequence number versus time trace for a TCP connection operating in the test environment, with a bit-error rate of $1x10^{-7}$. Over the course of this approximately 4,000,000 byte transfer, one TCP segment was corrupted. As we can see from the figure, the segment is retransmitted at time $\cong$ 11 seconds. In examining TCP's response to this loss in more detail, we see that Figure A-19 looks essentially identical to Figure A-14, which illustrates TCP's congestion response in detail. Since TCP cannot distinguish between loss due to congestion and loss due to corruption, and it *must* respond to congestion to avoid Internet congestion collapse, there is no other choice.

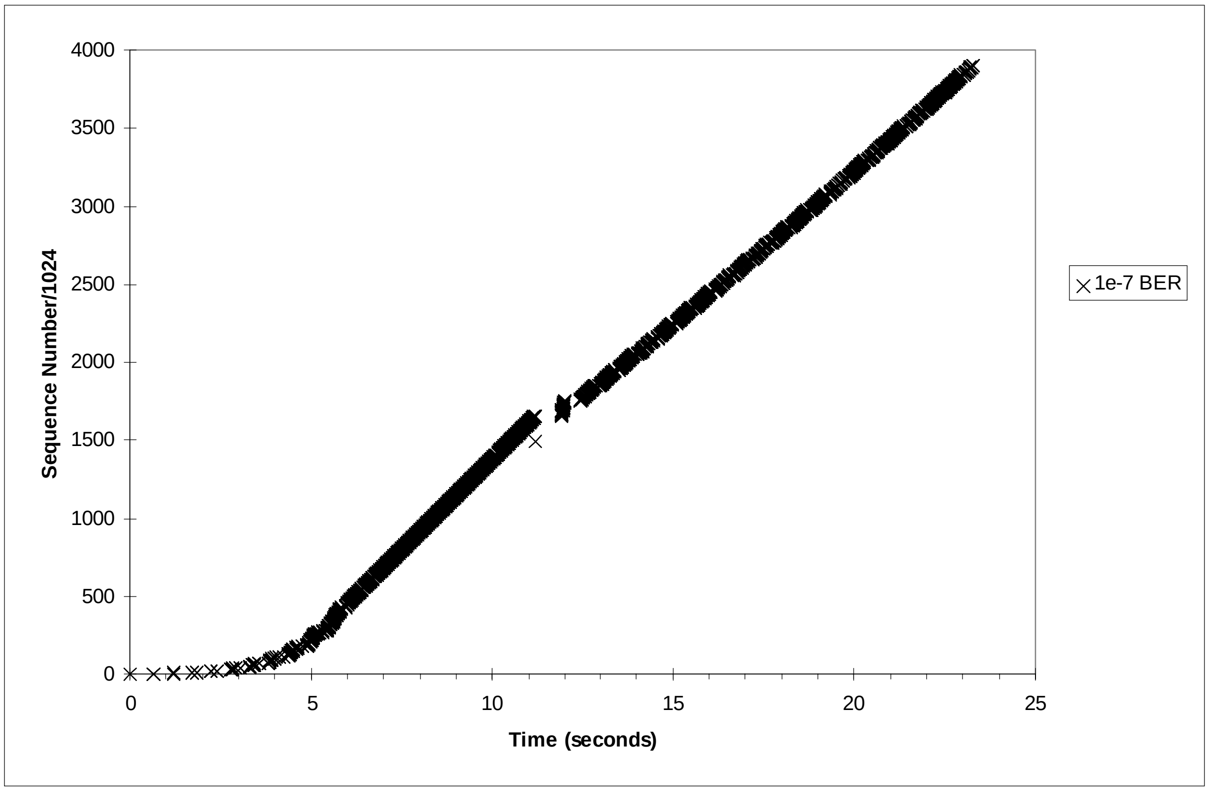


Figure A-18. Effect of One Bit-Error on TCP (BER = 1E-7)

Since the loss illustrated in Figure A-18 occurred earlier in the test than did the congestion loss of Figure A-13, we can illustrate one other aspect of TCP congestion avoidance. In TCP's congestion avoidance mode, TCP grows its congestion window by at most one segment per round trip. This results in a slow, steady increase in the congestion window, which, in an uncongested network, results in a slow, steady increase in transmission rate. If one lays a straight edge along the portion of the sequence number versus time trace that follows the loss (from time ≅ 12.5 through the end of the run), one can see that the line is not straight, but rather, it curves upward slightly. This corresponds to TCP increasing the congestion window gradually. (Approximately one packet per two round trips, or one packet per second, in this case.)

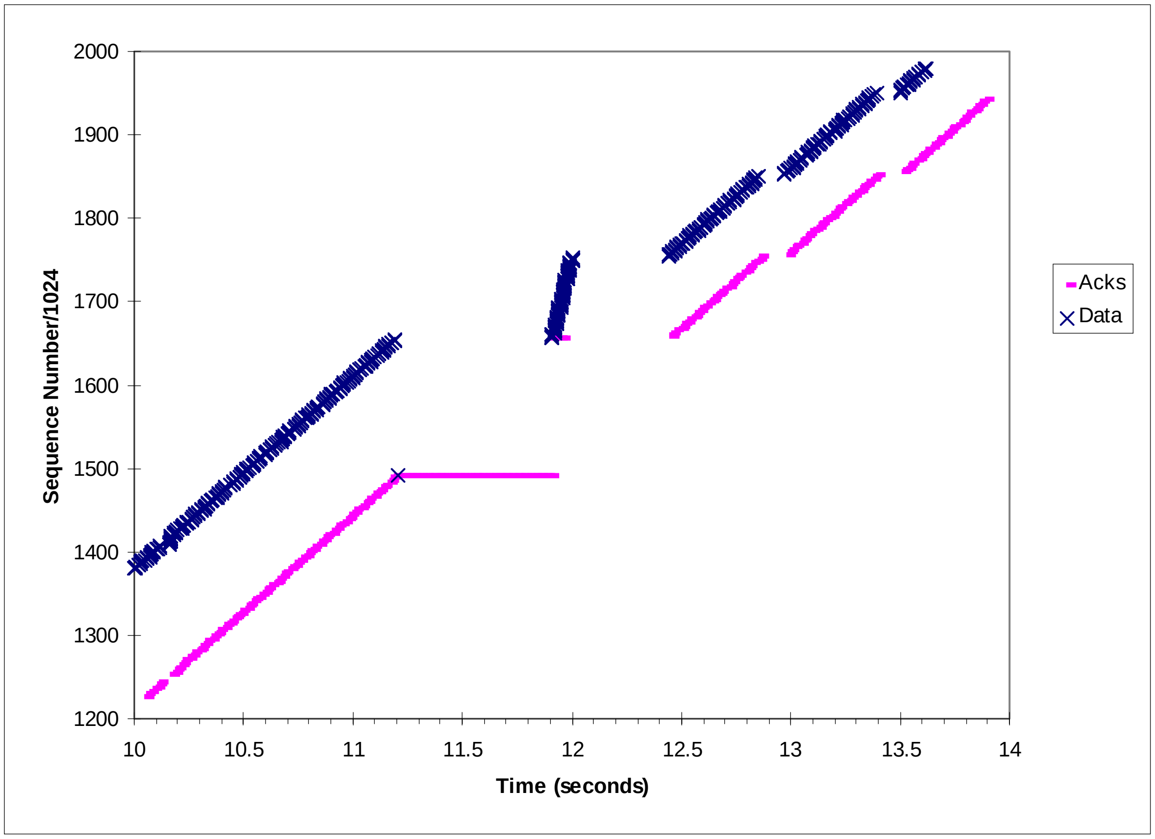


Figure A-19. Detail of TCP Error Recovery

Figure A-20 illustrates the response of TCP to two bit-errors. The measured bit-error rate on the link was approximately $6x10^{-7}$. The two errors occur within approximately three seconds of each other, so the congestion window is first halved, then halved again almost immediately. Recall that, because of the buffer memory imbalance, the congestion window is reduced to 60% of its previous value after the first loss. The second loss drops the connection to approximately 30% of its expected throughput. If this occurred late in the run, the effect would be minimal. The earlier in the run that the errors occur, the greater the impact. The worst possible case would be for an error or series of errors to occur early in slow start, because the connection would have a very small congestion window that it could only increase linearly through the congestion avoidance algorithm. (Refer to the excerpts from RFC 2001 in Section A.2.1 to see in detail why this is so.)

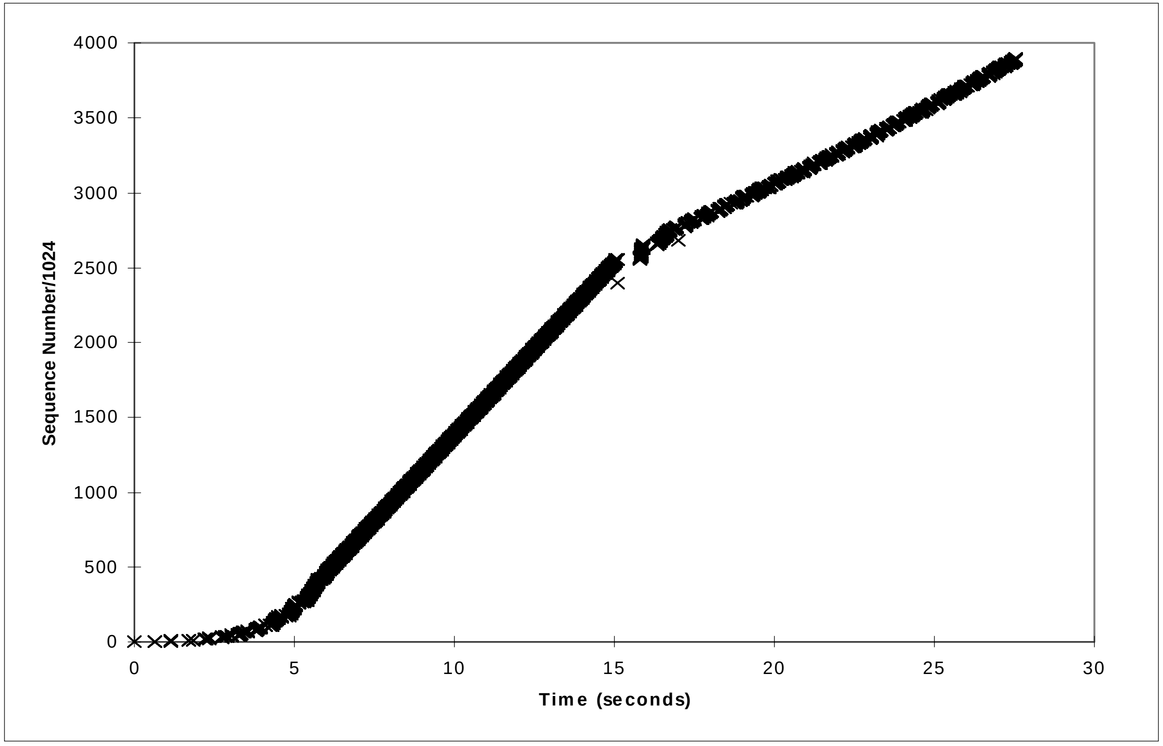


Figure A-20. Effect of Two Bit-Errors on TCP (BER = 6E-7)

Figure A-21 overlays the zero-error case with the one- and two-bit-error cases. Each of these traces is shown as a line, rather than a series of discrete points (hence the vertical and horizontal line segments in the region of the data losses). Recall that the data rate is 2,000,000 bps on the satellite link. As the data rate increases, the effect of a given bit-error rate becomes more severe, since the congestion window needs to be larger to fully utilize the capacity of the higher-rate links. It therefore takes many more round-trip times to grow the congestion window back to full use of the available capacity.

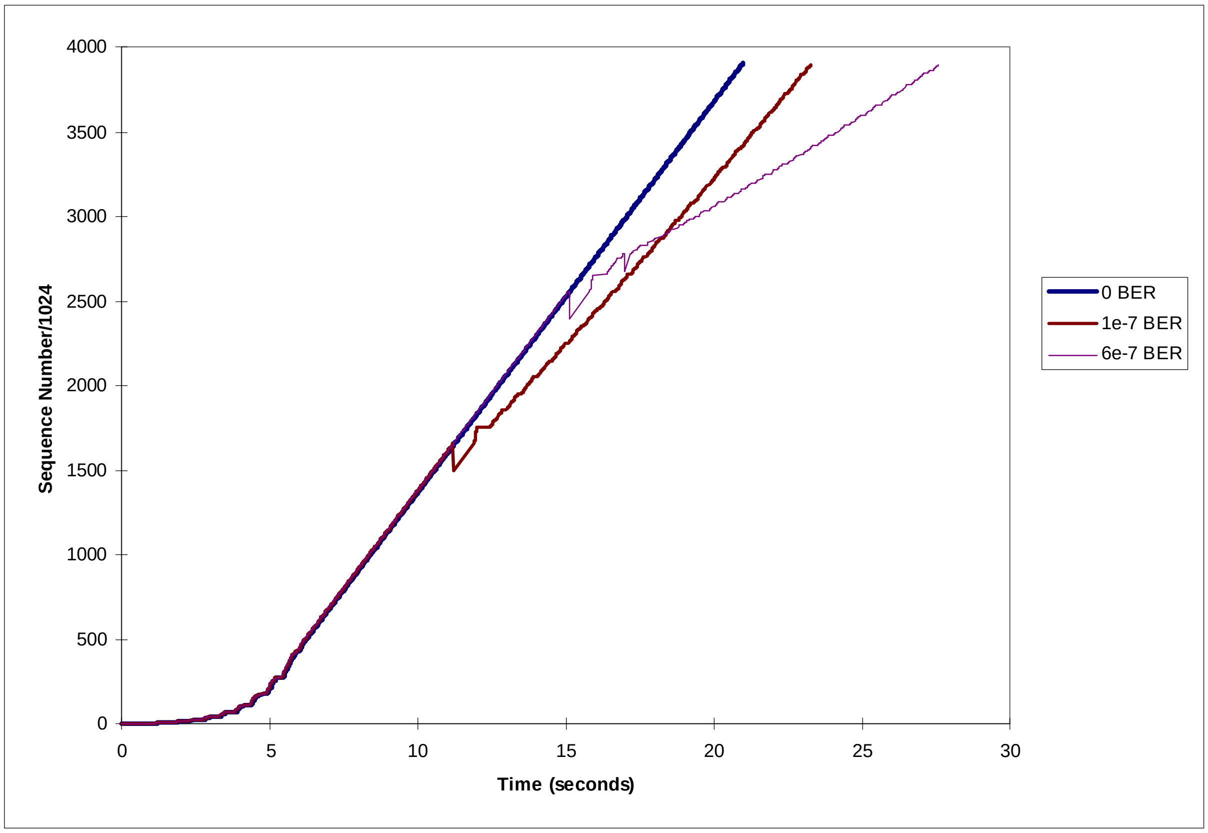


Figure A-21. TCP Performance with 0, 1, and 2 Bit-Errors

A.3.2 SCPS-TP Performance

We now consider the SCPS-TP response to corruption. Recall that we have modified the TCP-Vegas algorithm to *not* halve the congestion window in the event of loss (if a particular route is so configured, which it was for these tests). In addition to this, we have implemented a Selective Negative Acknowledgment (SNACK) option. The SNACK option allows the receiver to inform the sender of missing segments more effectively than TCP's duplicate acknowledgment method, and more bit-efficiently than the TCP Selective Acknowledgment (SACK) option documented in RFC 2018[3]. However, neither SACK nor SNACK are

[3] This statement is not intended in any way to diminish the potential usefulness of the TCP SACK option. We believe that the SACK option has the potential to be an excellent adjunct to the SNACK option, and that there is a place for both. The major plusses of the

congestion control mechanisms – they are both *data recovery* mechanisms [8]. Both SACK and SNACK provide TCP greater flexibility in identifying lost data. Standard TCP can only use its acknowledgment number, and must infer from duplicate acknowledgments that data is missing. Further, standard TCP has no idea how much data beyond the first segment is missing, so prudent practice indicates that only the first segment is retransmitted [RFC 2001]. Both SACK and the SCPS SNACK option allow missing data *beyond* the acknowledgment number to be identified. The SCPS implementation of TCP-Vegas congestion control allows us to interpret loss as something other than congestion. This results in the error performance summarized in the body of this document.

Figure A-22 illustrates a sequence number versus time trace for a SCPS-TP test operating with congestion control enabled, and a bit-error rate of $1x10^{-6}$. Both outgoing data and arriving acknowledgments are shown, and the packet size is 1452 bytes (1400 of which is user data. The retransmissions indicate that there were eight packets lost during the course of the run. (The retransmissions *also* indicate that there was a bug in our retransmission algorithm. Each missing segment is retransmitted twice, once when the duplicate acknowledgments or SNACK arrives, and again just before the retransmission is acknowledged. This is the unfortunate result of an experiment in persistent retransmission that we implemented, in which, once a segment is flagged for retransmission, it is retransmitted at a rate of once per round-trip time until it is acknowledged. The value of this algorithm is still somewhat in question – it has merit at very high error rates. However, our implementation of it in this test was flawed, in that our estimate of the round-trip time was too aggressive. We will see in Figure A-24 that this technique did provide some benefit, and may be appropriate for implementations that expect to operate in very high bit-error rates.) Once again turning our attention to Figure A-22, the important point to note is that congestion control is in operation, but is getting information about the congestion state of the network from queuing information, not from packet loss. The losses are treated as corruption, and are retransmitted (under both rate and congestion window control). As a result, the diminishment in overall throughput is negligible.

SACK option, including applicability to satellite communications, are documented in [Bruyeron, Fall, Mathis].

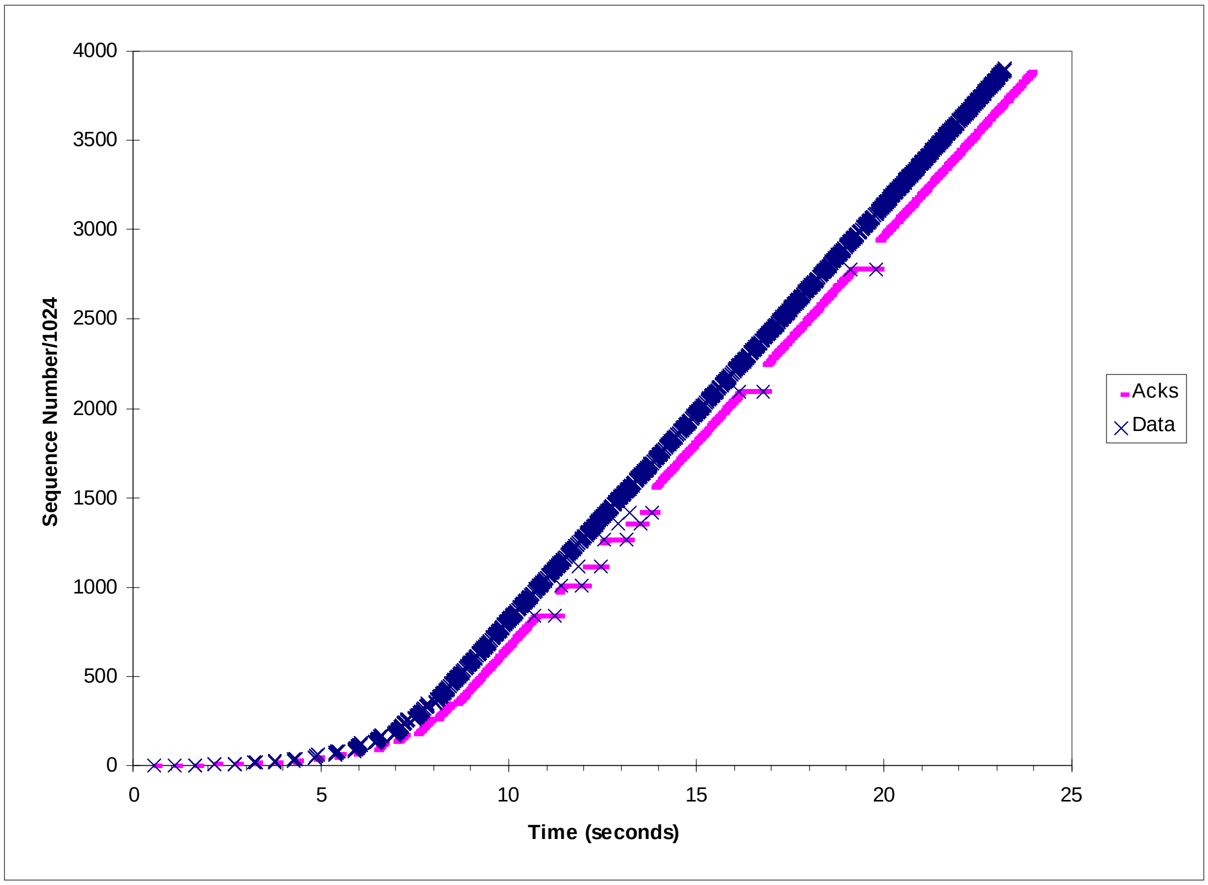


Figure A-22. SCPS-TP Corruption Response (BER = 1E-6)

Figure A-23 presents the response of SCPS-TP to a 1x10^{-5} bit-error rate. Over 50 segments required retransmission in this test. Through careful examination of the acknowledgments in relation to the data segments, we see that for this run at this error rate, there was no advantage to the aggressive retransmission policy – no retransmissions were lost. (We realize that the inadvertent multiple retransmission is, in fact, a *disadvantage*. However, we can tell from when the retransmissions are acknowledged whether a correct implementation of the scheme would have been useful.)

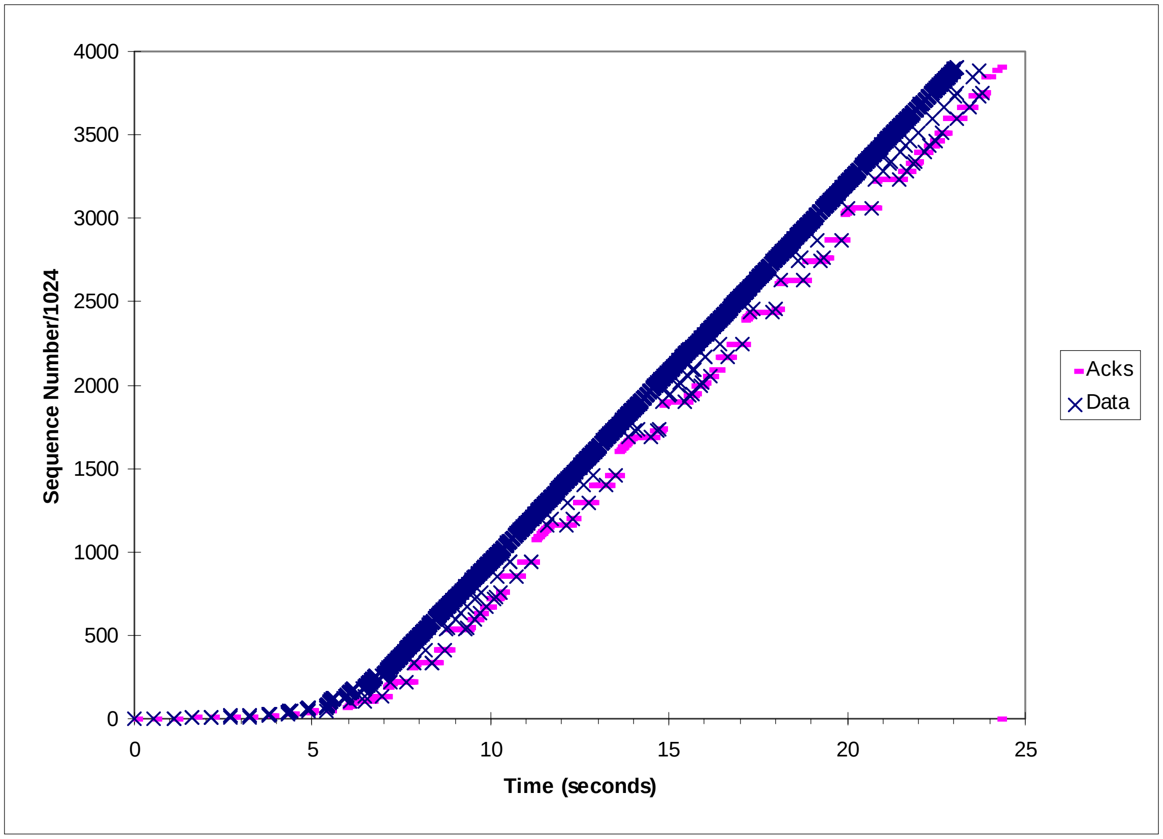


Figure A-23. SCPS-TP Corruption Response (BER=1E-5)

Finally, we consider Figure A-24, which is a SCPS-TP run at a bit-error rate of 5x10-5, with 1452-byte packets. If we refer to Figure 5 in the body of the document, we see that throughput has noticeably diminished at this error rate (from approximately 68% of the link capacity to approximately 58%). We also see in this figure that the multiple-retransmission scheme finally shows some benefit. At times ≅ 11, 15.5, 17, 19.8, 21.5, and 25 seconds, we see that the initial retransmission is not acknowledged, but the second retransmission is.

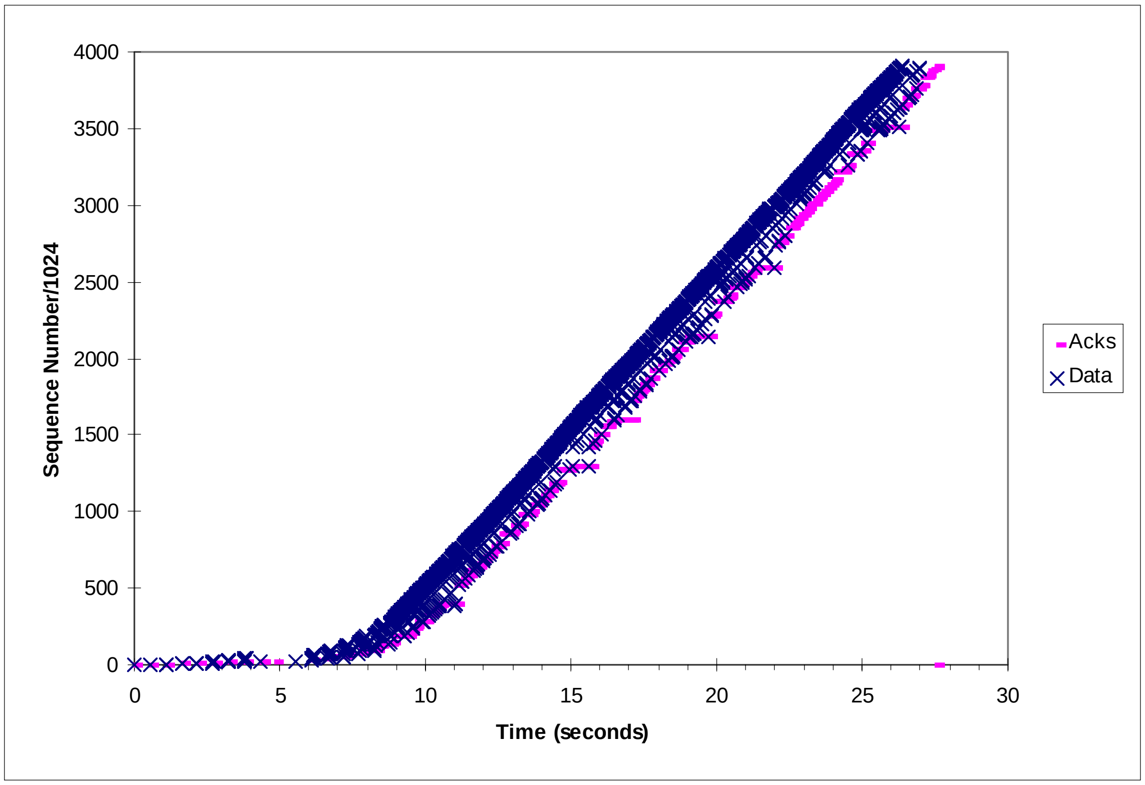


Figure A-24. SCPS-TP Corruption Response (BER = 5E-5)

Glossary

ACTS	Advanced Communication Technology Satellite
BER	Bit error rate
DOD	Department of Defense
DSCS	Defense Satellite Communication System
DES	Data Encryption Standard
FP	File Handling Protocol
FTP	File Transfer Protocol
FLTSAT	Fleet Satellite Communication System
GN	Ground node
IP	Internet Protocol
ISO	International Organization for Standardization
JTA	Joint Technical Architecture
LAN	Local area network
MD	Message Digest
NASA	National Aeronautics and Space Administration
NSA	National Security Agency
NLSP	Network Layer Security Protocol
NP	Network Protocol
OSI	Open Systems Interconnect
SCPS	Space Communication Protocol Standards
SMC	Space and Missiles System Center
SP3	Security Protocol Layer 3
SATCOM	Satellite Communications
SP	Security Protocol
SGLS	Space-Ground Link System
STRV	Space Technology Research Vehicle
SN	Space node
SNR	Signal-to-noise ratio

TP	Transport Protocol
TCP	Transmission Control Protocol
TT&C	Tracking, Telemetry, and Command
TPS	Test Preparation Sheets
WS	Workstation