

**Enhancement of the NMSU Channel
Error Simulator to Provide
User-selectable Link Delays**

by

Stephen Horan and Ru-Hai Wang

NMSU-ECE-00-001

NMSU-ECE-00-001

ENHANCEMENT OF THE NMSU CHANNEL ERROR SIMULATOR TO PROVIDE USER- SELECTABLE LINK DELAYS

Stephen Horan and Ru-Hai Wang

*Manuel Lujan Space Tele-Engineering Program
New Mexico State University
Las Cruces, NM*

Prepared for

NASA Goddard Space Flight Center
Greenbelt, MD

under Grant NAG5-7520

April 30, 2000



**Klipsch School of Electrical and Computer Engineering
New Mexico State University
Box 30001, MSC 3-O
Las Cruces, NM 88003-8001**

Contents

List of Figures	iii
List of Tables	iv
Summary	v
Introduction	1
Methods, Assumptions, and Procedures	2
Design Goals	2
Virtual Instrument Design Modifications	3
Rate-Splitter and Delay VI Components	7
Virtual Instrument Validation	11
Test Configuration	12
Results and Discussion	14
Protocol Transfer Initialization	14
TCP/IP fp Tests	18
SCPS fp Tests	21
Conclusions	24
References	25
List of Symbols, Abbreviations, and Acronyms	26

List of Figures

Figure Number	Caption	Page
1	Relationship between data source and data sink locations and the rate-splitting and delay modules	4
2	Computer arrangement for the channel error VI and the rate change and delay VIs	5
3	Front panel user interfaces for the modified data rate splitters. Each interface has user input for the delay amount.	8
4	(a) out-going forward link with incoming return link (b) out-going forward link with incoming forward link	9 10
5	tcpdump results for a 1-KB file transfer with a 10-second link delay	15
6	tcpdump results for a 10-KB file transfer with a 10-second link delay	16
7	ftp statistical throughput variation as a function of link delay and channel BER with symmetric transmission rates	19
8	ftp statistical throughput variation as a function of link delay and channel BER with unsymmetric transmission rates	19
9	ftp symmetric link variation with set-up delay added onto the statistical variation	20
10	ftp unsymmetric link variation with set-up delay added onto the statistical variation	20
11	fp statistical throughput variation as a function of link delay and channel BER with symmetric transmission rates	22
12	fp statistical throughput variation as a function of link delay and channel BER with unsymmetric transmission rates	22
13	fp symmetric link variation with set-up delay added onto the statistical variation	23
14	fp unsymmetric link variation with set-up delay added onto the statistical variation	23

List of Tables

Table Number	Caption	Page
1	Typical Path Delays (one-way)	2
2	Error Vector Files Used in SGLS Transmission Tests	12
3	Initial File Transfer Delay as a Function of Link Delay and Link Configuration	16
4	Total File Transfer Time as a Function of Link Delay and Link Configuration	17

SUMMARY

This is the third in a continuing series of reports describing the development of the Space-to-Ground Link Simulator (SGLS) to be used for testing data transfers under simulated space channel conditions. The SGLS is based upon Virtual Instrument (VI) software techniques for managing the error generation, link data rate configuration, and, now, selection of the link delay value. In this report we detail the changes that needed to be made to the SGLS VI configuration to permit link delays to be added to the basic error generation and link data rate control capabilities. This was accomplished by modifying the rate-splitting VIs to include a buffer to hold the incoming data for the duration selected by the user to emulate the channel link delay.

In sample tests of this configuration, the TCP/IP `ftp` service and the SCPS `fp` service were used to transmit 10-KB data files using both symmetric (both forward and return links set to 115200 bps) and unsymmetric (forward link set at 2400 bps and a return link set at 115200 bps) link configurations. Transmission times were recorded at bit error rates of 0 through 10^{-5} to give an indication of the link performance. In these tests, we noted separate timings for the protocol set-up time to initiate the file transfer and the variation in the actual file transfer time caused by channel errors. Both protocols showed similar performance to that seen earlier for the symmetric and unsymmetric channels. This time, the delays in establishing the file protocol also showed that these delays could double the transmission time and need to be accounted for in mission planning. Both protocols also showed a difficulty in transmitting large data files over large link delays.

In these tests, there was no clear favorite between the TCP/IP `ftp` and the SCPS `fp`. Based upon these tests, further testing is recommended to extend the results to different file transfer configurations.

INTRODUCTION

There exists a need for designers and developers to have a method to conveniently test a variety of communications parameters for an overall system design. This report is the third in a series describing the development of Virtual Instruments (VI) to configure an overall space channel simulator. The first two reports described, respectively, the Space-to-Ground Link Simulator (SGLS) VI [1] for channel bit error processing and the second VI [2] to provide a different data rate on the forward and return links on the simulated space channel. This third report describes the modifications to the rate change VI to allow for variable link propagation delays. All VI components are PC-based test devices programmed using the LabVIEW™ version 5 [3] software suite developed by National Instruments™. This instrument was designed to be portable and usable by others without special, additional equipment beyond multiple serial communications ports in the PC platforms.

This report describes the design goals for the VI module in the next section and follows that with a description of the design of the VI instrument. This is followed with a description of the validation tests run on the VI. An application of the delay test configuration to networking protocols is then given.

The design of the channel simulator is being tested with the standard TCP/IP File Transfer Protocol (ftp) and the Consultative Committee for Space Data System (CCSDS) Space Communications Protocol Standard (SCPS) File Protocol (fcp). These tests look at a standard commercial product and a similar protocol that has been optimized for space channel applications to investigate their performance similarities and differences. These are the same protocols that were used in the earlier series of tests ([1] and [2]).

METHODS ASSUMPTIONS AND PROCEDURES

DESIGN GOALS

The design goals for the overall space channel link simulator development were covered previously in [1] and [2]. The design of the link propagation delay modifications was intended to give the basic channel simulator the ability to emulate the propagation delays out to that found at a “Triana” orbit or 5 seconds on each link segment. Typical link delays for various satellite missions are given in Table 1. In this design, the propagation delay should be user-selectable to allow for maximum flexibility. To maintain symmetry in the module development and to allow for future enhancements, the link propagation delay was added as it would be found in nature: an individual delay for each leg and not a bulk delay for the end-to-end system.

Path	One-way Length (km)	One-way Delay
LEO	750	2.5 ms
GEO	35,700	120 ms
Lunar	384,000	1.28 s
Triana		5 s

This capability to model link delay is then added to the basic capabilities of independent error processes for the forward and return links and independent data rates for the forward and return links. While the delays on a satellite-based link are usually symmetric, they do not have to be. For example, tests have been run with ACTS [4] where a forward link was through the GEO satellite but the return was over land lines. This created a different link delay in each direction. To allow for maximum independence in link configuration, allowing the user to independently select the link delay in each direction is a design goal for the modification.

VIRTUAL INSTRUMENT DESIGN MODIFICATIONS

In this section, we develop the design for the modifications to the Virtual Instrument that provides for the ability of the channel error simulator to allow the user to select the desired amount of link delay on the forward and return links. This will include a description of the rate splitting methodology so that the user can see how the delay fits into the overall context. The error generation module is independent of this modification and the rate splitting function so that part of the simulator will not be described here. A full description of the LabVIEW software modules to perform the necessary functions is also given.

Method of Rate Splitting and Delay Generation

The rate splitting and delay generation methodology used in the VI is based the following data flow:

- a. Data will enter and leave the overall simulator (rate and delay functions as well as the error generation module) at the highest data rate to be used on either the forward or return data links.
- b. Two additional PCs will be used in conjunction with the SGLS error-generation PC to perform the rate splitting and delay functions independently on the forward and return data links.
- c. The additional PCs will drop the low-rate channel baud rate prior to entering the SGLS channel error simulator and then raise it back to the original level upon coming out of the SGLS and prior to passing it out of the overall simulator. These same PCs will provide the desired link delay on the forward and return data links using user-selectable delay values.

This overall process is illustrated in Figure 1 showing the relationship between the data lines, the splitting functions, and the delay functions. Here, *Tx* and *Rx* are the source and sink nodes for the data. *External* is the data link between the *Tx* and *Rx* nodes, and the overall space channel simulator. *Forward* is the path from *Tx* to *Rx* while *Return* is the path between *Rx* and *Tx* inside

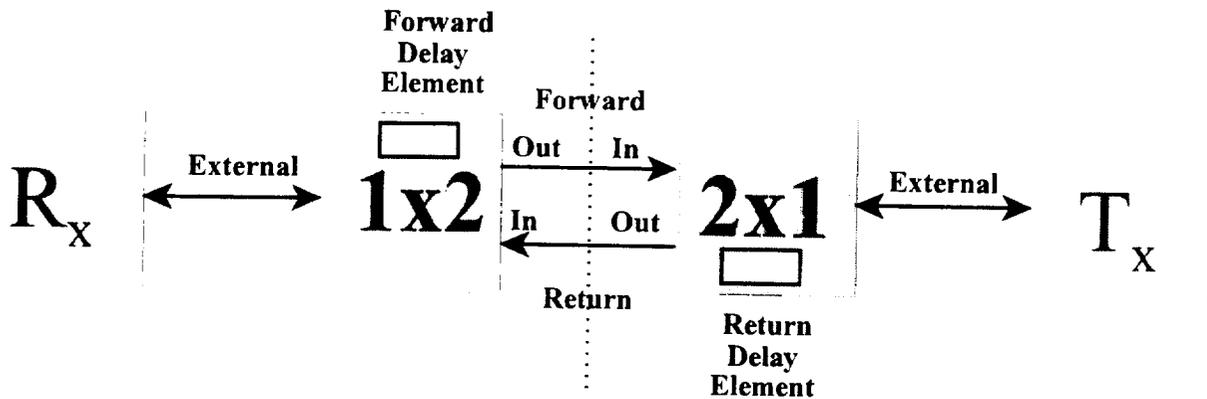


Figure 1 - Relationship between data source and data sink locations and the rate splitting and delay modules.

the channel simulator. The module 1×2 provides the interface between the *Forward* and *Return* paths inside the simulator, and the *External* link to the R_x node. The module also provides the *Forward* link propagation delay element. The 2×1 module provides the complementary interface between the overall simulator and the T_x node and the *Return* link propagation delay element. The link speeds and delays inside the 1×2 and 2×1 modules are independently selectable by the user. The error generation process for the forward and return links would be placed at the location of the vertical dashed line in the center of Figure 1. The LabVIEW VIs for the rate splitting and delay do not perform any error processing of the data streams. The actual hardware configuration for the computers and software modules is illustrated in Figure 2. Here, R_1 and R_2 are the forward and return data rates, the channel error PC hosts the SGLS VI for producing the channel errors, the rate change PCs host the VIs for splitting and recombining the separate data transmission rates and providing the link propagation delays. The channel error PC requires a total of four serial communications ports and each of the rate change computers requires a total of three serial communications ports.

Selection of VI Methodology

The Virtual Instrument was designed using the LabVIEW™ programming architecture. LabVIEW was chosen for the following reasons:

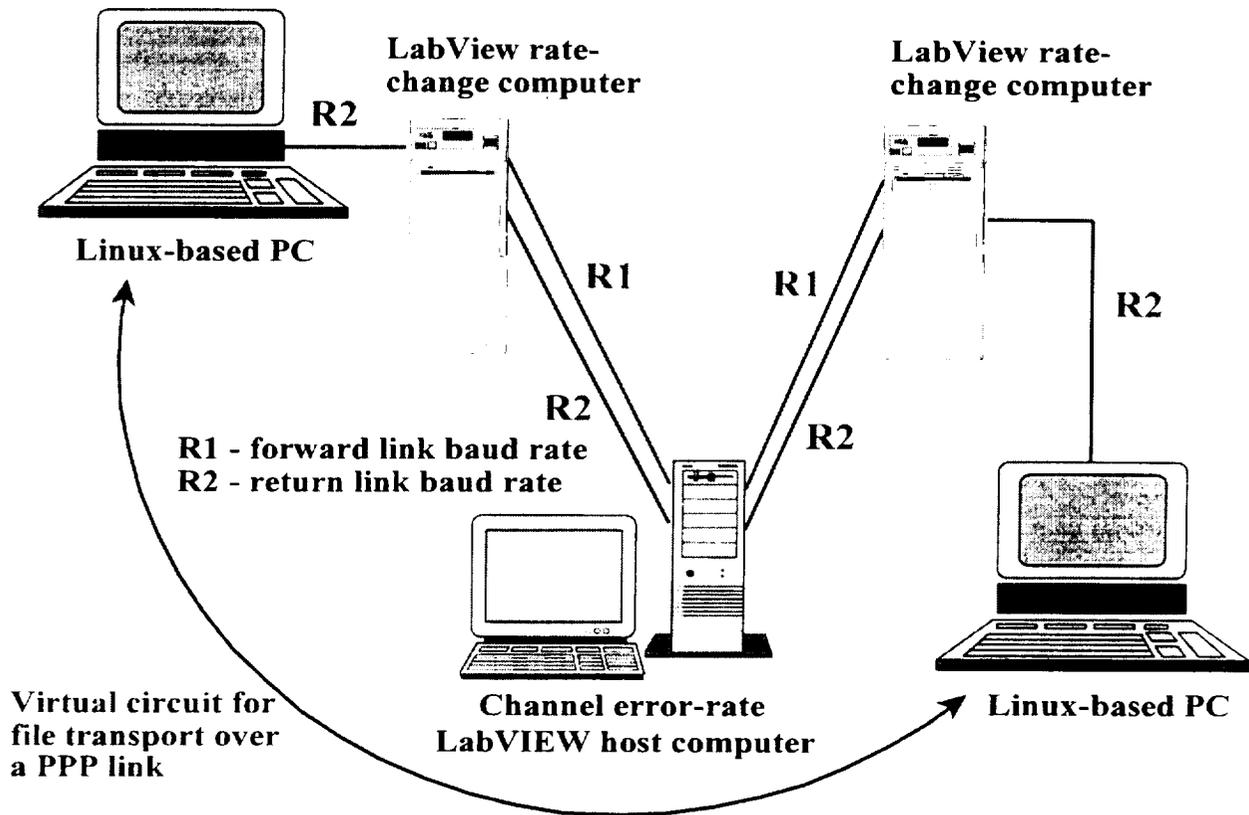


Figure 2 - Computer arrangement for channel error VI and rate change and delay VIs.

- a. The programming language is available on PC, Macintosh, and UNIX platforms;
- b. The programming language is object-oriented and allows for modular code development;
- c. The programming language provides for convenient access to PC communications ports (RS-232 and Ethernet) for data flow through the modules;
- d. This will allow development based on the initial SGLS development in LabVIEW to proceed using similar techniques.

LabVIEW is a graphical programming language that is data driven and not strictly sequence driven (it only operates on data as it becomes available). Additionally, LabVIEW manages all memory and I/O functions that normally the high-level language programmer would need to manage through programs and drivers.

The LabVIEW programming amounted to providing the necessary modifications to the existing VIs for the rate splitting and buffering operations. No additional modification of the existing VI for channel error generation functions were required to allow it to operate with the delay modifi-

cations. The VI rate-splitting and delay modules then provide the following capabilities using the programming language primitives and built-in modules:

- a. Allow for data flow in two directions simultaneously,
- b. Buffer the input links in both directions to prevent loss of data,
- c. Allow user-selectable link transmission rates for both data flow directions,
- d. Allow user-selectable link delay times for both data flow directions, and
- e. Use standard communications ports for data flow.

The general operation of the rate-splitting and delay VIs proceeds along the following steps:

- a. The user selects the appropriate VI to run for each of the two rate-splitting PCs based upon the data flow direction;
- b. The user initializes each VI and sets port numbers for data input and output (baud rate, delay value, and port number);
- c. The VI reads each directional serial port to determine if there is a data group present for pass through;
- d. If there is a data group available, that data group is read from the port and added to a circular buffer for the delay, and the number of bytes read and the time of the data arrival at the port are placed into arrays keeping track of the buffer contents;
- e. After the port-read operation, the current time is compared with the arrival time of the next data group in the circular buffer;
- f. If the required delay has passed, the VI strips the data group from the circular buffer and routes the data to the appropriate directional data port with the specified transmission rate, and clears number of bytes associated with the data group in the array tracking the circular buffer;
- g. The VI continuously loops as quickly as possible (no wait states: if no data are available at the input port, loop back and poll again) to move the data with minim processing delay.

There were no additional hardware modifications necessary to the simulator configuration to support the addition of the delay modules as there were for the rate-splitting functions.

RATE-SPLITTER AND DELAY VI COMPONENTS

The VIs have two parts to it: the user interface and the programming language. In this section, we will describe the details of both components. Consulting the LabVIEW programming manuals may be necessary if the reader is not familiar with LabVIEW concepts.

User Interface

The user interfaces for the rate-splitting and delay VIs provide the following features:

- a. Provide the user with a means to select the communications port for the forward and return data links and the communications port interfacing with the external computer system that will either source or sink the data. For these modules, the three RS-232 communications ports in the computer are used. The user decides which port is to be used for the input and the output of channel link to the SGLS and one port for the link to the data computer. LabVIEW designates COM1 as port 0, COM2 as port 1, etc. on the PC platform.
- b. Provide the user with a means to select the baud rate for the forward and return links. Normally, standard RS-232 rates will be selected. Most PC communications ports support baud rates from 1200 bps through 115200 bps.
- c. Provide the user with a means to select the link delay, in milliseconds, for the forward and return data links.
- d. Provide the user with real-time indications of data flow. This is done by showing the input queue size on each communications port upon each program iteration.
- e. Provide the user with a run-time means to disable the software processing (an on/off switch).

The user interface for the two VI components is illustrated in Figure 3. The input for the baud rate and delay values are provided using the LabVIEW Text Tool on the panel. The specification of the external data computer (External Port Number), and SGLS interface ports (Channel

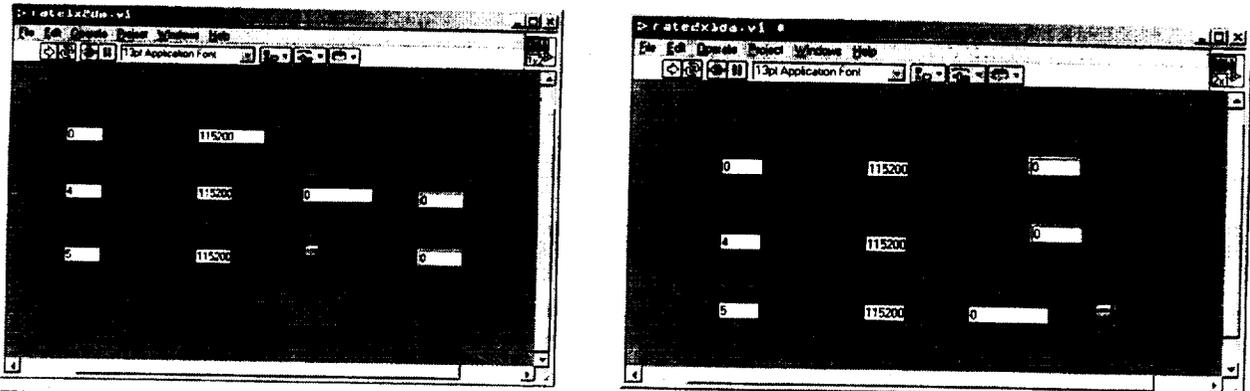


Figure 3 - Front panel user interfaces for the modified channel data rate splitters. Each interface has a user input for the delay amount.

Forward Port Number and Channel Return Port Number) are also selected using the LabVIEW Text Tool on the panel. The software enable/disable is done using the button switch marked *Continue* on the VI panel. This can be clicked by using the mouse to halt processing. The LabVIEW execution is initiated by clicking the left-pointing arrow (\leftarrow) on the command bar using the mouse.

VI Programming

The LabVIEW program is divided into two sections: module initialization and the processing loop as illustrated in Figure 4. During the initialization phase, the user input is taken from the VI front panel and is passed to the serial port control elements. This includes setting the external, forward, and return communications port numbers, the link delay value, and the communications baud rate for each port. The serial port initialization assumes the following communications port parameters to be in place and not changed by the user or the data sending device:

- a. 8 data bits, 1 stop bit and no parity bits on each byte transferred,
- b. No flow control is to be used, and
- c. A null modem cable will be used to connect to the serial port (a straight-through cable will not work properly).

After setting the communications ports and the delay value, the VI may be started and data flow

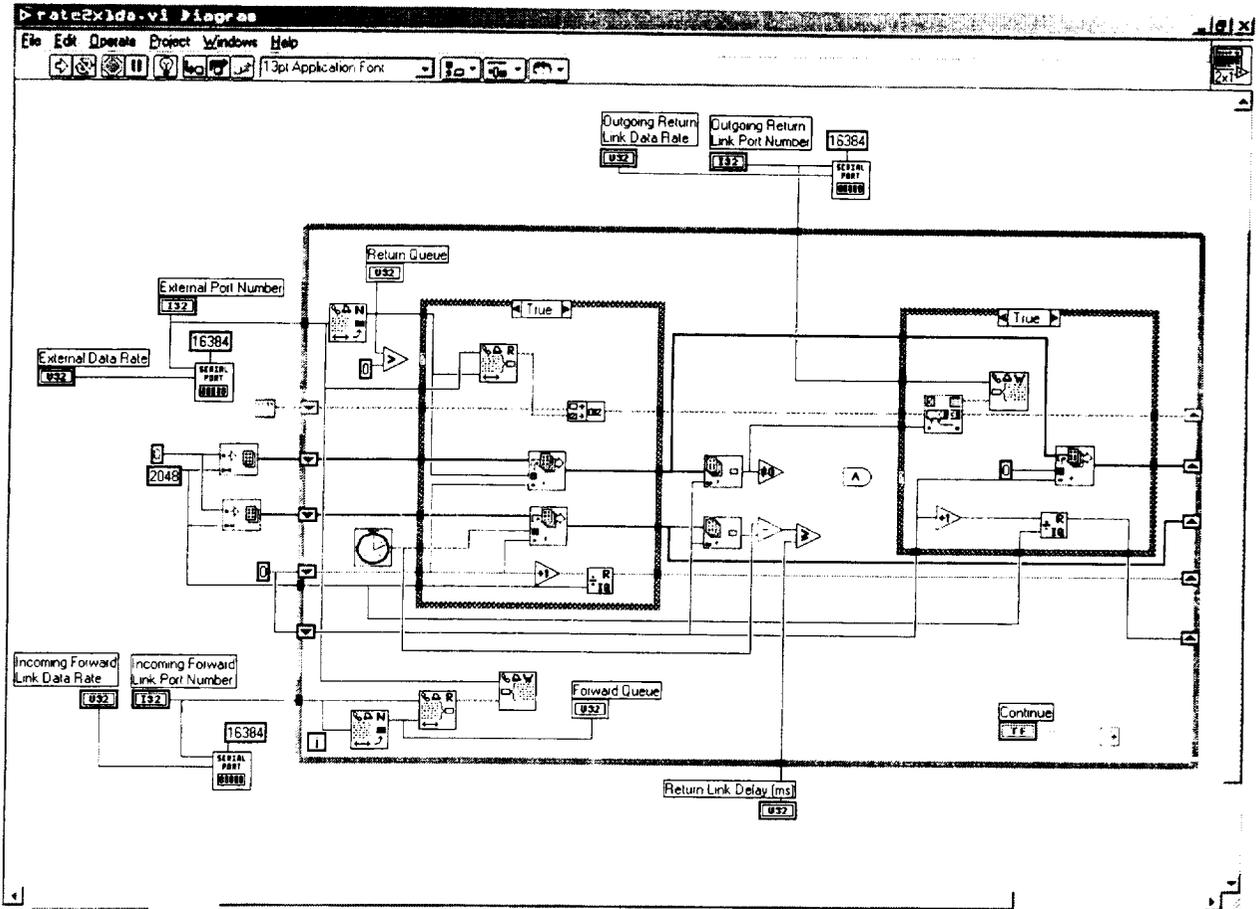


Figure 4 - (a) out-going forward link with in-coming return link

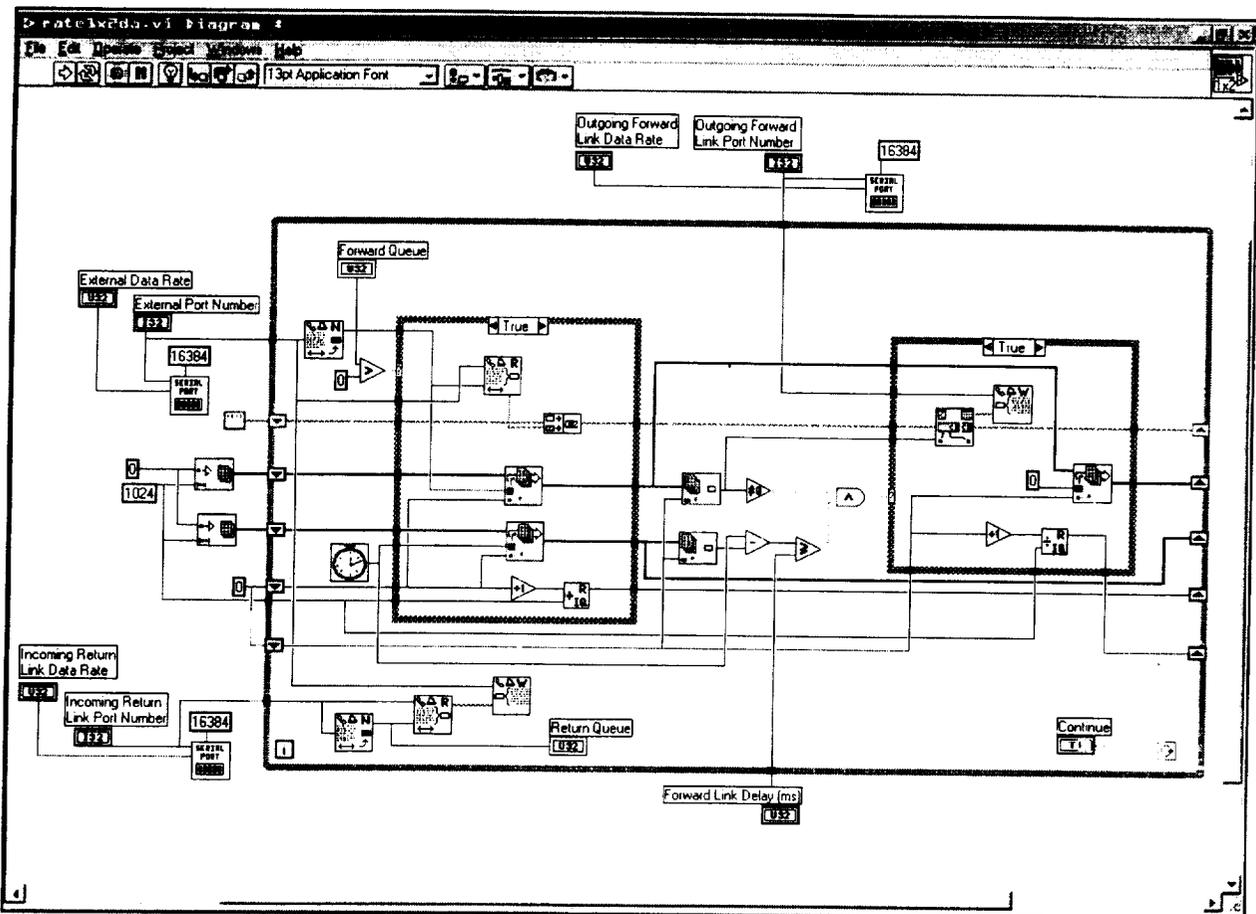


Figure 4 - (b) out-going return link with in-coming forward link.

may begin. The processing is controlled using a While Loop structure with no timing breaks and with continuous operation as long as the front panel button is not pushed. The processing loop proceeds as follows:

- a. Each input communications port is queried to determine if at least one byte of data is available (the loop only processes integer byte multiples of data) for routing;
- b. For each port, if the port has no data to be processed, nothing is done for that port on the loop iteration;
- c. If the port has data to be routed, all of the available bytes are read into the VI and are routed to the indicated output port if the data is “pass through” data (not subject to the delay element in this module);
- d. If the port has data to be routed and that data is subject to link delay in this module, all of

the available bytes are read into the VI and placed into the circular buffer; next the circular buffer is then checked to see if the next data group is ready for release (has waited the required delay duration) and if so, the data are routed to the indicated output port;

- e. The While Loop then starts the next iteration.

Processing will continue until the user either places the toggle switch on the VI front panel at the OFF position using the Operating Tool or when the user clicks on the LabVIEW stop button with the mouse.

VIRTUAL INSTRUMENT VALIDATION

The basic instrument validation is performed by working with each component of the VI as a self-contained sub module and using the VI display interface options to place debug displays at each step of the way. With these debug options in place, the data flow was monitored for correct operation. Typical debug tests included:

- a. Validation of the passing of data through the individual serial ports as individual characters are entered from the keyboard and re-running the XMODEM tests to ensure proper passing of file data;
- b. Monitoring the input queue size to verify that it did not exceed 16384 bytes at which point data can be lost;
- c. Verification of correct data flow when interfaced with the SGLS VI on the SGLS PC;
- d. Running ping tests to verify that the single-link and dual-link delay times were as expected based upon the round-trip time reported by ping.

After verifying data flow was proceeding as expected, ftp tests were conducted to ensure correct data flow and connections were properly made with TCP/IP.

TEST CONFIGURATION

The tests run with the SGLS delay configuration were conducted using the computers as was shown in Figure 2. The source and destination computers for the file transfer were (a) a Gateway PC with a 133 MHz processor speed and 16 MB of memory and (b) a Dell PC with a 266 MHz processor speed and 128 MB of memory. Both were running Red Hat Linux version 6.1. The Gateway computer is of a similar class to that expected in a small satellite while the Dell computer is more similar to that expected in a ground station configuration. The file source and sink computers were connected to the test computers using commercially-obtained 6-foot null modem cables. The three VI PCs were connected using the J10/DB-9 straight-through cables obtained from National Instruments terminated with a null modem connector and a gender changer. The SGLS computer was configured with a National Instruments 4-port serial card with J10 interface ports. The two rate-change PCs were configured with a National Instruments 2-port serial card. These additional serial cards were obtained to give the additional serial ports necessary for this configuration. Tests were run at channel Bit Error Rates (BER) of 0 through 10^{-5} using the files listed in Table 2 which were the same files used in the basic SGLS test. Files to be transferred were selected from the same set of random text files having lengths of 1 KB, 10 KB, 100 KB, and 1000 KB used in the SGLS testing. For each file transmission test, ten runs were performed and the average time to complete the transmission recorded. These are noted in the file results.

BER	Vector File
0	infinite.dat
10^{-6}	a1075d.dat
10^{-5}	a975d.dat

There were two series of tests run in this initial set of experiments. Each test consisted of ten trials at a given BER and link delay. In the first series of tests, the transmission rate in both the

forward and return directions was set to 115200 bps (symmetric link tests). In the second series of tests, the transmission rate in the forward direction was set to 2400 bits per second and 115200 bits per second (unsymmetric link tests) in the return direction. In both series of tests, the ftp and fp protocols were examined while the BER was varied over 0, 10^{-6} , and 10^{-5} . The one-sided link delay was set to 3 ms, 120 ms, 1280 ms, or 5000 ms. A fixed file size of 10 Kbytes were used in these tests.

In the following sections, the results for these tests are given for each protocol. The results will show both the statistical variation due to channel errors and the statistical variation added to the protocol initialization delay.

RESULTS AND DISCUSSION

In this section, we look at the results of the tests run to verify the correct operation of the SGLS with the channel delay enhancements. The results are divided into three major areas: protocol transfer initialization, TCP/IP `ftp` test results, and SCPS `fp` test results. The results are discussed and/or compared in each area.

PROTOCOL TRANSFER INITIALIZATION

The first set of tests was to scope out the reactions of the Internet protocols to delay times. During these tests, we determined that we needed to document both the file transfer times plus the management overhead times. In previous testing, all of the link processing and propagation delays were well under 100 ms so the protocols did not show much effect. Now, with round-trip link delays up to 10 seconds being possible, this separate protocol management delay needs to be accounted for as well. The reason becomes apparent in the following plots for `ftp` file transfers using the TCP/IP protocol. For example, using the `ftp` report for the delay time, one would see that the time to transmit a 1-KB file, when transferred with a 10-second round trip delay under the condition of no transmission errors, as taking under 1 millisecond to complete. However, the process time as measured by the observer watching the transfer from the computer console from the time the command is entered until `ftp` reports that it is finished is closer to 30 seconds. The results of the record of the transaction using the `tcpdump` utility are given in Figure 5. Here, the actual file transfer takes place along the time occupied by the vertical line with the **FIN** at the top. The 1-KB file is transferred in a single packet and the transfer time is less than 1 millisecond as reported by `ftp`. The actual transfer begins back at time 0 seconds on the plot. The three diamonds marked **SYN** show the protocol management packet transfers leading up to the actual file transfer. Finally, all of the protocol acknowledgments are completed at around a time of 20 seconds. This total time is unreported by `ftp` to the user but it is an important part of the process for determining the actual time to complete a transfer through a long link delay.

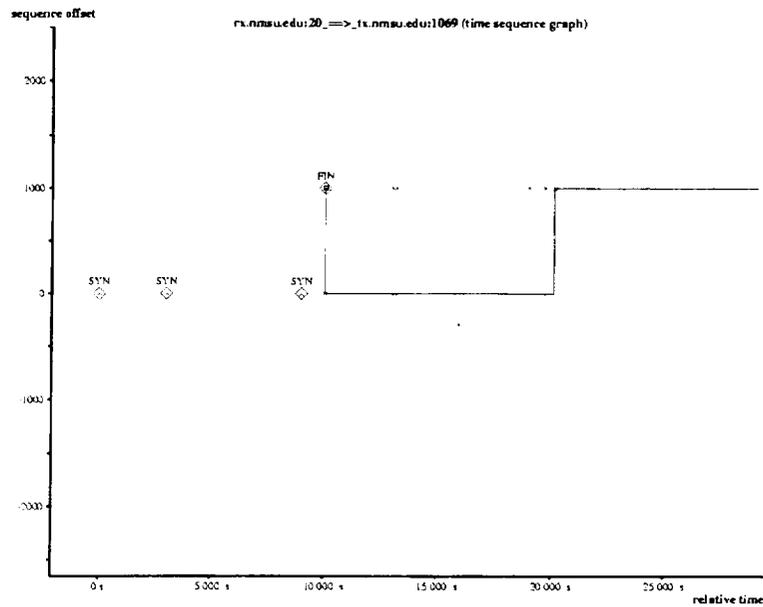


Figure 5 - tcpdump results for a 1-KB file transfer with a 10-second link delay.

This management overhead time is reasonably consistent as can be seen with the 10-KB file transfer result shown in Figure 6. Here, we see the same protocol management overhead at the start of the transfer prior to the actual data transfer (the three **SYN** marks between seconds 0 and seconds 10 on the plot). The `ftp` process reports a transmission time of approximately 20 seconds for this file transfer that occurs between seconds 10 and seconds 30 on the plot, but the management time is not included in the `ftp` process report.

Table 3 shows a summary of the time delay between the start of the file transfer process and the actual transfer of data beginning based on averaging three sample runs. This time is the management time to negotiate the transfer. The table shows the results for transferring a 10-KB file using both the TCP/IP `ftp` service and the SCPS `fp` service under both symmetric (115200 bps) and unsymmetric (115200/2400 bps) link conditions. The link delay mentioned in the Table is applied to both the forward and return links. From Table 3, we see that the set-up time is directly proportional to the link delay with a linear offset. We can approximately model the set-up

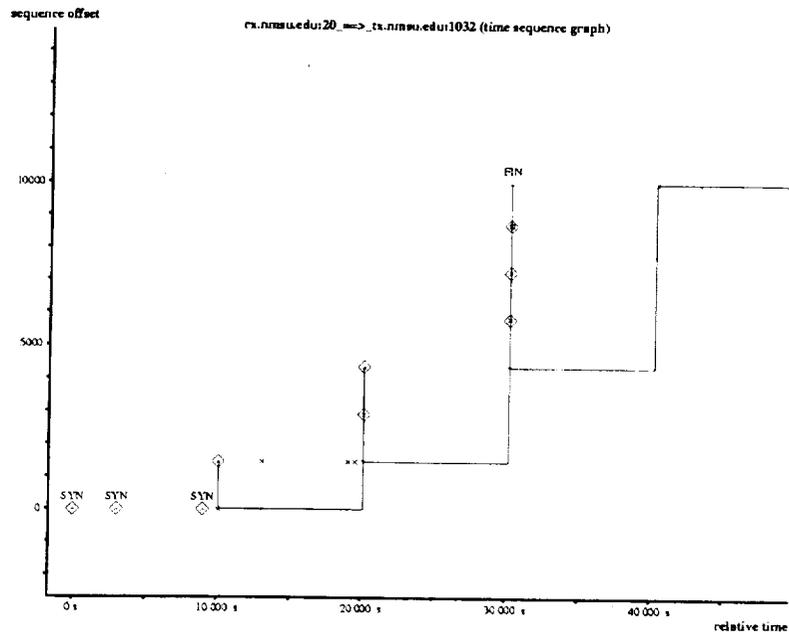


Figure 6 - tcpdump results for a 10-KB file transfer with 10 second link delay.

Table 3. Initial File Transfer Delay as a Function of Link Delay and Link Configuration.			
Protocol	Link Delay	Symmetric Link	Unsymmetric Link
FP	3 milliseconds	60 milliseconds	522 milliseconds
	120 milliseconds	303 milliseconds	755 milliseconds
	1280 milliseconds	2.64 seconds	2.97 seconds
	5000 milliseconds	10.03 seconds	10.46 seconds
FTP	3 milliseconds	54 milliseconds	450 milliseconds
	120 milliseconds	295 milliseconds	709 milliseconds
	1280 milliseconds	2.61 seconds	3.01 seconds
	5000 milliseconds	10.04 seconds	10.46 seconds

time as

$$Setup_Time = Link_Delay + Processing_Delay + SGLS_Delay$$

where the *Processing_Delay* will be a function of the host computer activity and will be somewhat random. The *SGLS_Delay* is the time to propagate the signal through the simulator. Using the ping utility, this is approximately 30 ms. The sum of the *SGLS_Delay* and the *Processing_Delay* will be approximately 50 ms. The difference between the Symmetric and Unsymmetric columns comes from the fact that the links have unbalanced data rates in the latter case so the protocol handshake packets effectively see different link delays in the two directions. The entire file transfer time including the initial protocol handshaking as well as the end-of-transfer protocol handshaking is given in Table 4. This total time is greatly in excess of the time reported for the file transfer in the ftp reporting and was derived from the tcpdump timing of the file transfer exchange between the SYN and FIN markers (see Figures 5 and 6 for examples of these markers). This total time is similar to that a user would notice for the file transfer. The total times are of the same order of magnitude for both the ftp and fp protocols. The management time to establish the transfers were nearly the same for both protocols as seen in Table 3. However, there are differences in the total time as given in Table 4 with the advantage going to the fp protocol in most cases.

Table 4. Total File Transfer Time as a Function of Link Delay and Link Configuration.			
Protocol	Link Delay	Symmetric Link	Unsymmetric Link
FP	3 milliseconds	276 milliseconds	1.32 seconds
	120 milliseconds	933 milliseconds	1.74 seconds
	1280 milliseconds	5.6 seconds	6.33 seconds
	5000 milliseconds	20.23 seconds	20.89 seconds
FTP	3 milliseconds	187 milliseconds	1.15 seconds
	120 milliseconds	641 milliseconds	1.66 seconds
	1280 milliseconds	5.23 seconds	9.21 seconds
	5000 milliseconds	30 seconds	31.3 seconds

In running these tests, even without errors on the channel, we found that the 1000 Kbyte files would not transmit to completion when the link delay was 5000 milliseconds in each direction (10 seconds total delay). This was true for both the `ftp` and `fp` transfer methods. Therefore, for the error performance tests below, this file size was not attempted.

TCP/IP FTP TESTS

The first series of tests performed was the transmission of files using the TCP/IP `ftp` service with the channel error rates and the transmission rates mentioned above. The statistical variation portion of these tests is summarized in Figures 7 and 8 where the transmission times for the 10 Kbyte file are displayed as a function of BER and link delay. The total transfer time consisting of the transfer set-up time from Table 3 and the statistical variation from these test runs is illustrated in Figures 9 and 10. Interesting items noted during these tests include:

- a. The basic symmetric and unsymmetric link variations as a function of BER are found with the added link delay as well.
- b. Symmetric and unsymmetric tests at low channel delay values ran a bit faster than earlier tests reported in [1]. We attribute this to using a significantly faster computer both in the SGLS channel simulator and as part of the data transfer computer configuration.
- c. The unsymmetric link at long delay values (5 seconds each link) runs at nearly the same throughput as does the symmetric link thereby reducing the effect of the data transfer imbalance. The link delay does have a relatively greater effect for the small link delays.

From these results, we can see that both the BER and the link transfer rates affect the overall throughput performance. The management set-up time needs to be included in the link transfer time. This management set-up time is about twice the round-trip link delay except on the shortest links where the operating system and computer delays dominate.

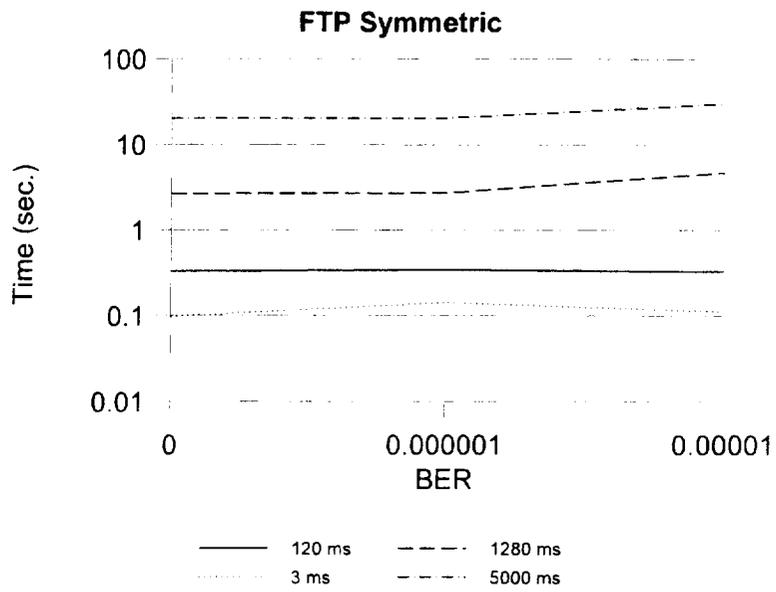


Figure 7 - ftp statistical throughput variation as a function of link delay and channel BER with symmetric transmission rates.

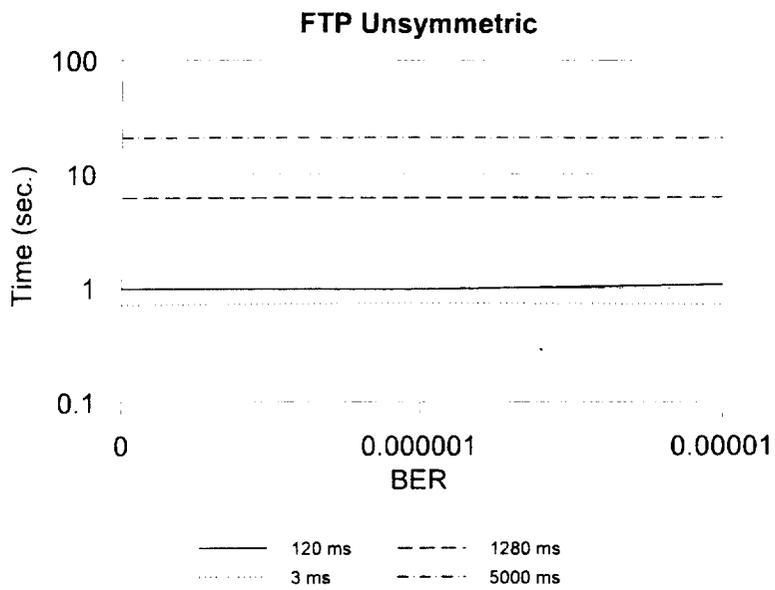


Figure 8 - ftp statistical throughput variation as a function of link delay and channel BER with unsymmetric transmission rates.

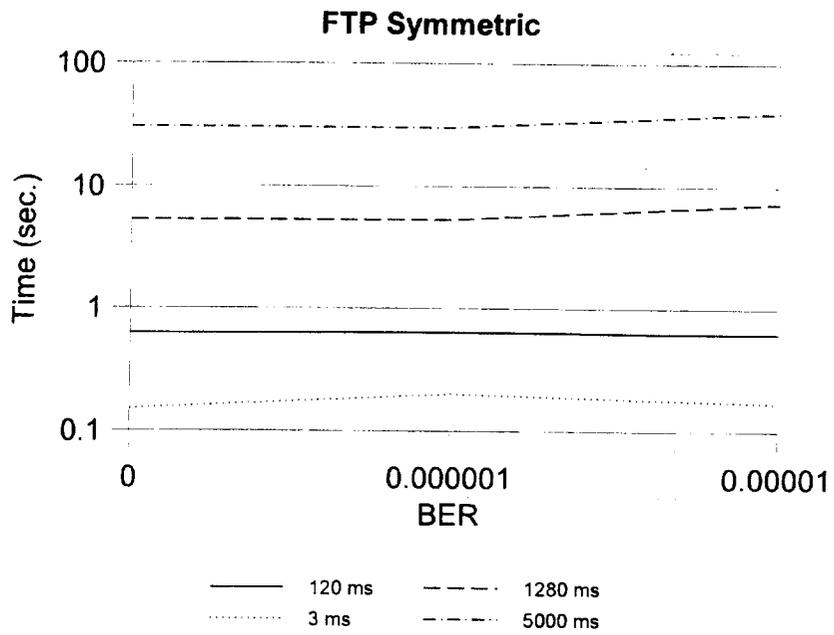


Figure 9 - ftp symmetric link variation with set-up delay added onto the statistical variation.

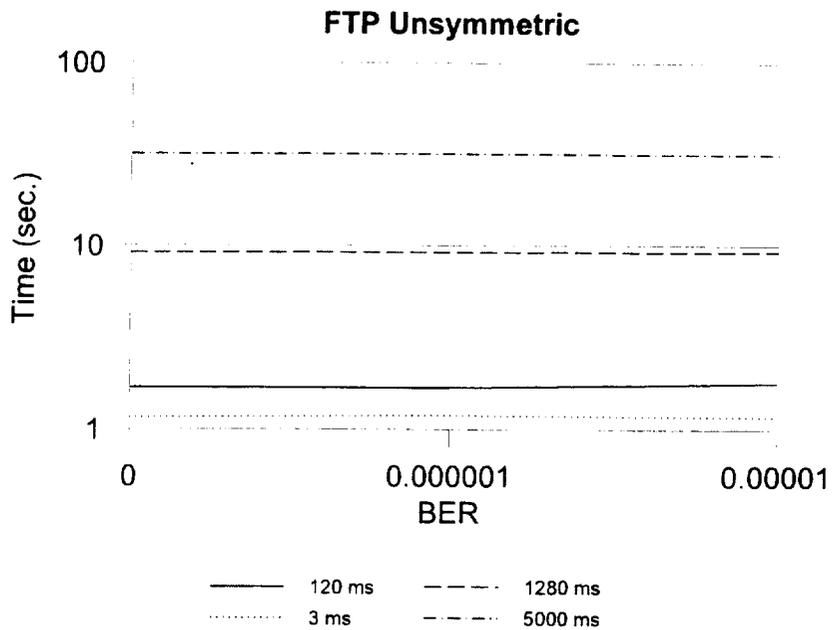


Figure 10 - ftp unsymmetric link variation with set-up delay added onto the statistical variation.

SCPS FP TESTS

The second group of tests performed was the transmission of files using the Consultative Committee for Space Data Systems (CCSDS) Space Communications Protocol Specification (SCPS) File Protocol (fp) service [3] with the same channel error rates and the transmission rates used in the TCP/IP ftp tests. Version 1.1.8 of the SCPS reference implementation provided by MITRE [4] was used. The statistical variation portion of these tests is summarized in Figures 11 and 12 where the transmission times for the various file sizes are displayed as a function of BER and link transmission rates. As was done with TCP/IP, the total time consisting of the transfer set-up time from Table 3 is added to the statistical variation from these test runs and the result is illustrated in Figures 13 and 14. Interesting items noted during these tests include:

- a. Generally, the SCPS fp ran faster than did the TCP/IP ftp for large link delays while for small link delays, the two protocols ran at about the same throughput.
- b. The SCPS fp appears to be more sensitive to the channel errors than does ftp during these tests. With only 10 runs per test, some of this may just be small-number statistics in the test program.
- c. The SCPS fp shows the same speed-up, as did ftp, in the throughput over results reported earlier for the case of short channel delays which is attributed again to using faster computers in the test configuration.

As in the case of ftp, we can see that both the BER and the link transfer rates affect the overall throughput performance. Again, the management set-up time needs to be included in the link transfer time. This management set-up time is about twice the round-trip link delay except on the shortest links where the operating system and computer delays dominate.

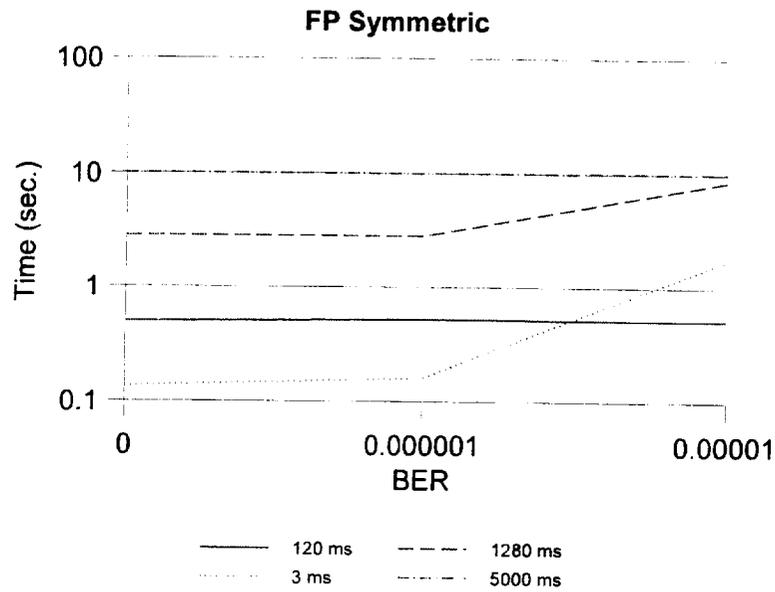


Figure 11 - fp statistical throughput variation as a function of link delay and channel BER with symmetric transmission rates.

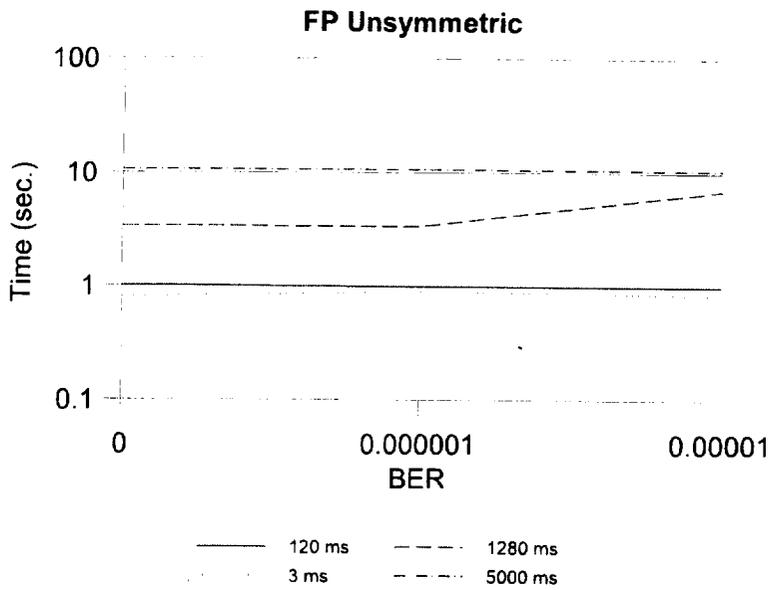


Figure 12 - fp statistical throughput variation as a function of link delay and channel BER with unsymmetric transmission rates.

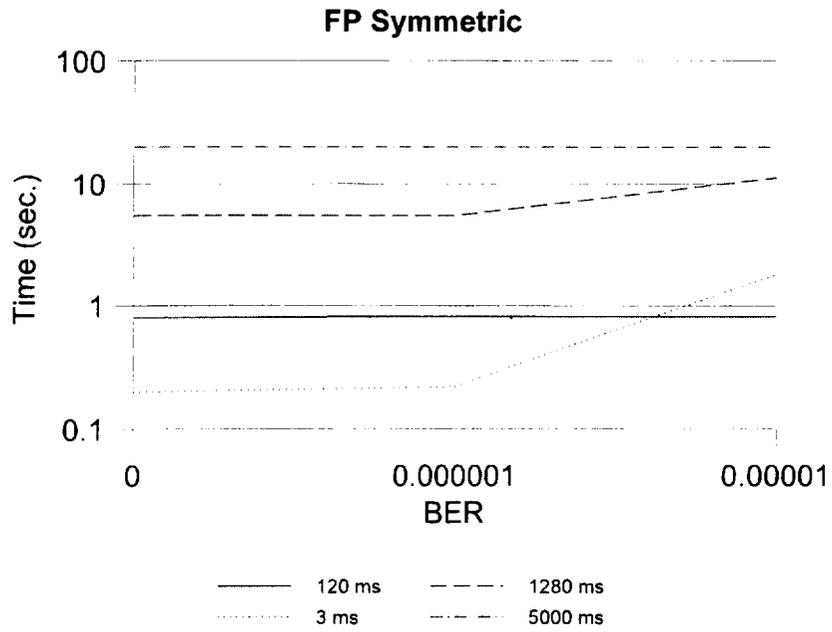


Figure 13 - f_p symmetric link variation with set-up delay added onto the statistical variation.

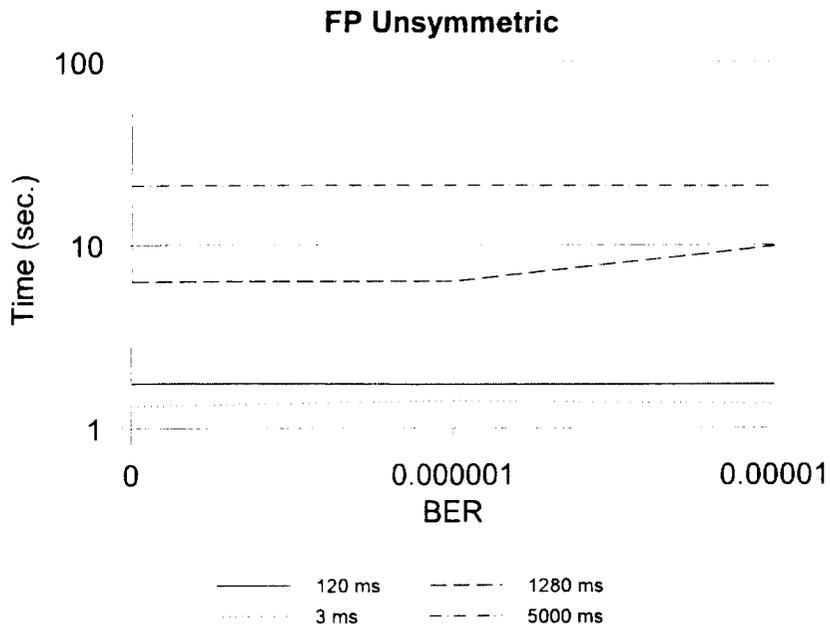


Figure 14 - f_p unsymmetric link variation with set-up delay added onto the statistical variation.

CONCLUSIONS

From these tests designed to verify correct operation of the SGLS modules, we draw the following conclusions as a guide for further testing:

1. Long transmission delays can impede the transmission of long files. In these tests, we were unable to complete 1 MB file transfers with link delays of 5 seconds on each path. While these long delays are not normal for near-earth applications (they are intended to model Triana-type communications links), the result does show that care needs to be taken in matching link delay to file size.
2. The link errors still interact with throughput measurements for both the SCPS and TCP/IP protocols.
3. Both the SCPS and TCP/IP protocols need approximately two round-trip hop times to initialize their transmission protocols. This needs to be factored into the anticipated link availability time when planning transmissions.
4. There was not clear favorite in these limited transmission tests between the SCPS `ftp` and TCP/IP `ftp` methods for transmitting the files. On some runs, one did better than the other but overall across all configurations, there was no clearly-preferred method.

Naturally, these conclusions are based on a limited set of runs used to determine the correct operation of the enhanced SGLS simulation environment. Further testing will be conducted to verify and extend these results.

REFERENCES

- [1] Horan, S. and Wang, R, "Design of a Channel Error Simulator Using Virtual Instrument Techniques for the Initial Testing of TCP/IP and SCPS Protocols," NMSU-ECE-99-002, 1 April 1999.
- [2] Horan, S. and Wang, R, "Enhancement of the NMSU Channel Error Simulator to Provide Unbalanced Forward and Return Transmission Rates," NMSU-ECE-99-003, 8 April 1999.
- [3] National Instruments™, LabVIEW™, Austin, Texas, January 1998.
- [4] Horan, S., Deny, M.C., Anderson, K.S., and Fowler, R.J., "Real-Time Control of Remote Sites: Using the ACTS with the Apache Point Observatory," NASA ACTS Results Conference, Cleveland, OH, September, 1995.
- [3] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification (SCPS) - File Protocol (SCPS-FP)," CCSDS 717.0-R-3, September 1997.

ACRONYM LIST

ACTS	Advanced Communications Technology Satellite
AWGN	Additive White Gaussian Noise
CCSDS	Consultative Committee for Space Data Systems
fp	File Protocol
ftp	File Transport Protocol
NMSU	New Mexico State University
PC	Personal Computer (Intel/Windows based configuration)
SCPS	Space Communications Protocol Standard
SGLS	Space-to-Ground Link Simulator
TCP/IP	Transmission Control Protocol/Internet Protocol
VI	Virtual Instrument