

An Evaluation of Protocol Enhancing Proxies and Modern File Transport Protocols for Geostationary Satellite Communication

Patrick E. Finch
University Corporation at Monterey Bay
NASA Ames Research Center
Moffett Field, CA 94035
605-604-4324
patrick.e.finch@nasa.gov

Donald V. Sullivan
NASA Ames Research Center
Moffett Field, CA 94035
605-604-0526
donald.v.sullivan@nasa.gov

William D. Ivancic
NASA Glenn Research Center
Cleveland, OH 44135
216-433-3494
william.d.ivancic@nasa.gov

Abstract—NASA is utilizing Global Hawk aircraft in high-altitude, long duration Earth science missions. Communications with the science payload is via Ku-Band satellites in geostationary orbits. All payload communications use standard Internet Protocols and routing, and much of the data to be transferred is comprised of very large files. The science community is interested in fully utilizing these communication links to retrieve data as quickly and reliably as possible. A test bed was developed at NASA Ames to evaluate modern transport protocols as well as Protocol Enhancing Proxies (PEPs) to determine what tools best fit the needs of the science community. This paper describes the test bed used, the protocols, the PEPs that were evaluated, the particular tests performed and the results and conclusions.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. PEP AND PROTOCOL TESTING.....	1
3. PROTOCOL DESCRIPTION.....	2
4. HYPOTHESIS	2
5. TESTS	2
6. TESTBED.....	3
7. TEST RESULTS	3
8. SUMMARY AND CONCLUSIONS	6
9. THE WAY FORWARD	7
REFERENCES.....	7
BIOGRAPHIES.....	8

1. INTRODUCTION

NASA has acquired two Global Hawk Unmanned Aerial Vehicles (UAVs) to enhance upper atmospheric science in support of the Earth Science Project Office (ESPO). Communication with the experimental payload is by a Ku-Band satellite link. NASA's Earth Science Technology Office (ESTO) Advanced Information System Technology (AIST) program funded the "Real-Time and Store-and-Forward Delivery of Unmanned Airborne Vehicle Sensor Data" task to improve the amount and type of science that can be performed via sensors onboard Unmanned Aerial Vehicles (UAVs). This will be accomplished by improving access to sensors, improving data gathering, increasing timely access to critical data and improving data throughput during times of connectivity. To accomplish this, NASA's Glenn and Ames Research Centers are improving the data throughput and utilization of current UAV remote sensing

by developing and deploying technologies that enable efficient use of the available communications links – in particular, deployment of efficient transport protocols that use Internet technologies

2. PEP AND PROTOCOL TESTING

The users of the Global Hawk atmospheric research platform are the scientists (i.e. the Principal Investigators and their collaborators). This group is interested in ease of use and maximum delivery of science data. Their preference is to use as many existing Internet protocols as possible. Doing so allows the scientists to test their instruments and data collecting in the lab, on the ground, and in flight using the same protocols, commands, and scripts. The PIs desire to use the exact same Internet tools used in the lab while controlling instrumentation onboard the Global Hawk while maintaining a good user experience. Here, a good user experience is getting the required science data down in a timely manner. This has to be done while operating over near-error-free links with 600 milliseconds Round Trip Time (RTT) delays.

The protocols currently used are all based on the Transmission Control Protocol (TCP). They include: Telnet, Secure Shell (SSH), and file transfer protocols (i.e. File Transfer Protocol (FTP), Secure Copy Protocol (SCP), Secure File Transfer Protocol (SFTP), RSYNC, WGET, etc...). Often the file transfer protocols are run in an SSH tunnel.

In the late 1980s through the 2000s, in the days of Windows® and Windows XP®, manual tuning of TCP parameters was required for data links with very large bandwidth/delay products (BDP) such as found in high bandwidth Geostationary satellite links. This manual tuning is non-trivial and does not scale (every end user had to be a networking expert). One solution - not without problems - was to put a protocol enhancing proxy (PEP) between the two end systems [1]. A PEP is used to improve the performance of the Internet protocols on network paths where native performance suffers due to characteristics of a link. PEPs generally attempt to improve the traditional TCP control loops by either spoofing TCP connections or breaking the TCP connections into two and placing a protocol between the TCP end systems that does not assume all packet loss is due to congestion. This allows the protocol tuning to be performed in the PEPs rather than in all the end systems passing data through the PEPs.

Over the years, TCP implementations have improved to include many self-tuning features as well as to eliminate or reduce some security vulnerabilities. Thus, the questions to be addressed are “Will a PEP improve system performance and if so, by how much?” or, “Should we simply use existing TCP-based protocols available in modern kernels or perhaps use a rate-based protocol?”

3. PROTOCOL DESCRIPTION

TCP Protocol

TCP is a reliable transport protocol built into the kernel of computer operating systems. It guarantees reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another. TCP is designed to operate over shared networks. It includes flow control and congestion control algorithms that work well in the terrestrial Internet. The congestion control algorithm design is such that operation over noisy and long delay links is problematic. TCP will guarantee data delivery, but the link will not be used efficiently as TCP assumes any loss to be caused by congestion. In particular, standard TCP often fails to fully utilize the network capacity in large links with large BDP due to the limitation in its conservative congestion control algorithm. Much work has been done and is ongoing to improve the connection’s throughput by adopting more aggressive loss-based congestion control algorithms – particularly in Linux builds [2]. One example is TCP-CUBIC, which is the default TCP implementation in Linux Kernel 2.6.19 [3]. Another example is Compound TCP (CTCP) developed by Microsoft, which is designed to provide good bandwidth scalability with improved RTT fairness, and at the same time achieves good TCP-fairness, irrelevant to the windows size [4].

Saratoga Rate-Based Transport Protocol

Saratoga is a simple, lightweight, content dissemination UDP-based protocol that operates at line rate or, for some implementations, at a fixed-rate setting [5]. Saratoga was designed for use on private networks with no competing traffic and is capable of transferring very large amounts of data reliably under adverse conditions. Current implementations of Saratoga assume no congestion and thus deploy no congestion control algorithms. On private links, where one can be assured there is no congestions, there is no need to probe the system to determine available bandwidth or to reduce data-rates when losses occur as all losses are assumed to be due to errors rather than congestion. Regardless, work is ongoing to add congestion control feature to Saratoga so it can be used over links with competing traffic [6].

A rate-based PERL implementation of Saratoga is used as one of the protocols to compare with PEPs.

4. HYPOTHESIS

Our evaluation of PEPs versus modern file transfer protocols was limited to our particular operational environment, which is a private network with no competing traffic and fully bidirectional links. The Global Hawk network is, for the most part, error-free, has a round trip time delay of approximately 600 ms and has a bandwidth of 8 Mbps. For completeness, we tested over a variety of delays and error-rates.

We hypothesized the following outcomes:

1. A PEP would provide no improvements for very command and control communications and interactive tasks using TCP as PEPs are really designed to improve throughput for bulk transfers.
2. The Rate-based Saratoga implementation would perform better than any file transfers using PEPs or modern TCP implementations as this implementation of Saratoga assumes no loss due to congestion and implements no congestion control.
3. In a large BDP environment, the PEP would perform as good as or better than TCP files transfers without a PEP.

5. TESTS

Our tests consisted of evaluating a number of TCP-base file transfer protocols with and without a PEP as well as telnet to simulate interactive and command and control traffic and Saratoga. Also, we where curious to see how a PEP would handle traffic over secure shell (SSH) as the users often SSH into the system then run applications over the SSH tunnel. Users also use Secure Copy (SCP), Secure File Transfer Protocol (SFTP) and remote synchronization (rsync). Thus, we tested all of these protocols with and without a PEP for delays of 0, 100, 300 and 600 millisecond and bit-error-rates of 0, 10^{-7} and 10^{-5} .

The Space Communications Protocol Specification (SCPS) PEP base on the SCPS Transport Protocol (SCPS-TP) [7] was used. SCPS-TP is a set of TCP options and sender-side modifications to improve TCP performance in stressed environments including long delays, high bit error rates, and significant asymmetries.

6. TESTBED

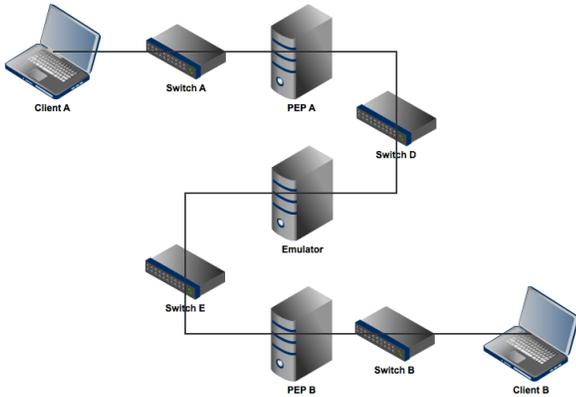


Figure 1: Testbed network diagram

The testbed was comprised of 5 computers and 4 switches (Figure 1). Client A, a CentOS 5.4 machine running Linux kernel 2.6.18, was in the role of a computer being used by a PI during a mission. Client A was using the Binary Increase Congestion control (BIC) algorithm [8]. PEP A, a CentOS 5.5 machine running Linux kernel 2.6.18, was in the role of a transparent gateway device, which sat between Client A and the greater network. Emulator, a Knoppix 4.0.2 machine running Linux kernel 2.6.18-rc4, was acting in place of a satellite link [9]. PEP B, a CentOS 5.5 machine running Linux kernel 2.6.18 was acting as the aircraft side transparent gateway. Client B, an Ubuntu 10.04 machine running Linux kernel 2.6.32 was acting as an instrument controller on board the aircraft. Client B was using the CUBIC congestion control algorithm [10], which is enhance version of BIC.

Client A was configured with a single Ethernet interface allowing it to get on to the 10.99.99.0/24 network as 10.99.99.2. PEP A, in acting as a transparent gateway was configured with two separate Ethernet interfaces, neither of which had an IP address. Emulator, in its role as a satellite link emulator also acted as a transparent gateway and was likewise configured with two separate Ethernet interfaces, neither of which had an IP address. PEP B, was also configured as a transparent gateway with two Ethernet interfaces and no IP addresses. Client B, in its role as an instrument controller, was configured with a single Ethernet interface with an IP address of 10.99.99.254 giving it access to the 10.99.99.0/24 network. All of the switching hardware was 100BASE-TX.

For our purposes, bandwidth was clamped on Emulator to a nominal 8 Mbps with a ceiling of 10 Mbps allowing 128 KB bursts of data. This was accomplished using the Linux command ‘tc’ using the Hierarchical Token Bucket queuing strategy. Specifically, we allowed 8 Mbps data rates for constant connections; but connections that are just starting can burst a total of 128 KB of data at line rate and any data sent after that will be throttled down to 8 Mbps. The ‘tc’ command is part of the iproute2 suite of tools that comes

standard with the NASA Channel Emulator Live CD used to emulate the satellite link during testing²[11], [12].

7. TEST RESULTS

Baseline Tests

TCP and UDP baseline tests were performed using nuttcp, a TCP/UDP network test tool [13].

The UDP tests provided a baseline for data throughput and confirmed our bandwidth settings in the testbed. The UDP test showed that when we configured our testbed for 8 Mbps throughput with 10 Mbps ceiling rate and 128k bytes burst rate the maximum sustainable throughput measured was 8.0069 Mbps.

TCP Baseline

All tests were run as single flow with full duplex links. The satellite emulator bandwidth was set to 8 Mbps providing a sustainable UDP throughput rate of 8.0069 Mbps. Packet size for all file transfer tests was 1500 Bytes with a 100 MB file transferred for each run³. For all tests through the SCPS PEP, the PEP was configured for a 400 ms one-way delay (800 ms RTT), 10 Mbps bandwidth with a minimum value for the TCP retransmission time set for 800 ms, the RTT [14].

Telnet Tests

The file transfer tests were run with the same settings as the TCP baseline tests except a large file was not transferred. The telnet tests were simply run to validate our initial assumption that a PEP would not noticeably improve the user experience for transactional communications. This was confirmed via user experience. Measurable data was not taken to support this simple experiment. Rather, the operator experience was the defining measurement.

File Transfer Tests

The file transfer tests were run with the same settings as the TCP baseline tests.

² # tc qdisc add dev eth0 parent 1: handle 2: htb default 10
tc class add dev eth0 parent 2: classid 2:10 htb rate 8mbit ceil 10mbit
cburst 128k
tc qdisc add dev eth1 parent 1: handle 2: htb default 10
tc class add dev eth1 parent 2: classid 2:10 htb rate 8mbit ceil 10mbit
cburst 128k

³ The transferred file was a 100 MB disk image.

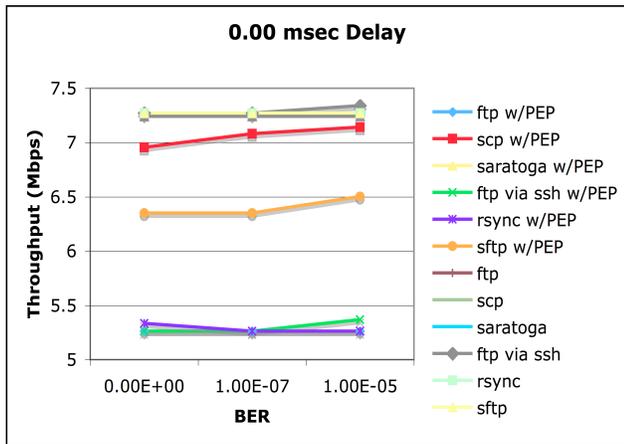


Chart 1: Throughput vs. BER for No Delay

Chart 1 Shows the performance of various file transfer protocols with and without a PEP for zero delay for various bit error rates (BERs). Note that performance with a PEP is worse than without a PEP. This is to be expected as a PEP simply adds overhead and processing for systems with low BDP. This is one end of the extreme.

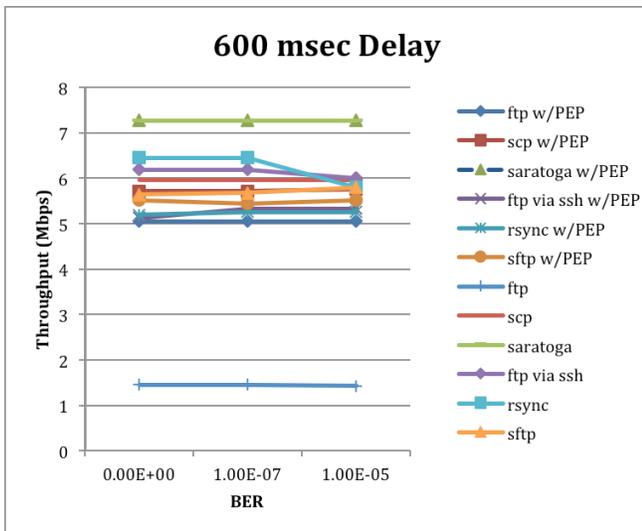


Chart 2: Throughput vs. BER for 600 ms RTT Delay

Chart 2 shows the other extreme. This graph shows performance results for a system with a RTT of 600 ms providing the largest BDP considered.

In chart 2 we see a PEP helping ever so slightly for TCP-based file transfers at 1.0 E-5 BERs. At near error-free, the PEP is not showing improvements. This is due to two factors: there are few losses and modern TCP aggressive congestion control schemes allow the congestion window to remain stable when only a few packets are lost as well as keeping the system from self-congesting. Causing self-congestion was the case with older forms of TCP where

once the bandwidth threshold was reached, the rate was halved and a very conservative linear increase ensured⁴.

In chart 2 one can see that generic ftp performs quite poorly over a 600 ms RTT delay even with no errors. This is due to FTP having a compiled in default buffer limiting the throughput performance to approximately:

$$\text{Throughput} = (\text{FTP Buffer}) / \text{RTT}$$

Chart 3 shows the performance of various file transfer protocols versus RTT delay for a BER of 1.0 E-5.

Looking at chart 3, one can see that Saratoga is consistent across delay and BER and has little fall off in performance at 1.0 E-5 BER. This is expected, as Saratoga is a negative acknowledgement (NACK) UDP-based protocol whose performance should be independent of delay as well as relatively low BERs. At high BERs, performance for any protocol will fall off as the packet losses increase and retransmissions increase. Furthermore, packet loss is related to the BER, the BER characteristics (i.e. bursty versus evenly distributed) and the packet length.

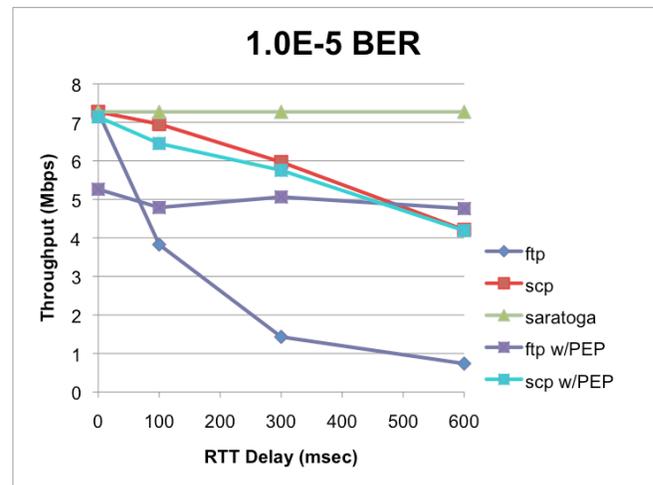


Chart 3: Throughput vs. RTT Delay for BER 10⁻⁵

Chart 3 also shows that a PEP improves the performance of FTP, but has little effect (and actually worsens) SCP for BERs up to 1.0E-5

Figures 2 through 9 are time-sequence plots of TCP file transfers over an 8 Mbps bandwidth and 600 ms RTT delay resulting in a BDP of 600,000 bytes. These plots were created using tcptrace [15], tcpdump [16], and xplot [17]. The markings on these plots indicate the following:

- Green Line keeps track of the ACK values received from the other endpoint.
- Yellow Line tracks the receive window advertised from the other endpoint. (It is drawn at the sequence number

⁴ This congestion avoidance algorithm is known as additive increase/multiplicative-decrease (AIMD).

value corresponding to the sum of the acknowledgment number and the receive window advertised from the last ACK packet received.)

- Little Green Ticks track the duplicate ACKs received.
- Little Yellow Ticks track the window advertisements that were the same as the last advertisement.
- White Arrows represent segments sent. The up and down arrows represent the sequence numbers of the last and first bytes of the segment respectively.
- Red Arrows (R) represent retransmitted segments with the up and down arrows similarly representing the sequence numbers of the last and first bytes of the segment.

Some of the important characteristics shown are the TCP window size (denoted by the vertical distance from the green line to the yellow line); packets sent (denoted by vertical white lines) and retransmitted packets (denoted by vertical red lines).

For the FTP transfer without a PEP as shown in figure 2, the advertised receive window size is approximately 12,000 bytes. Such a window size is far below the BDP of our link (i.e. 600,000 bytes). This is also the case for FTP between the client and the PEP [Figure 3]. The SCPS PEP was set for a window size of 2,000,000 bytes as recommended in the SCPS user guide.

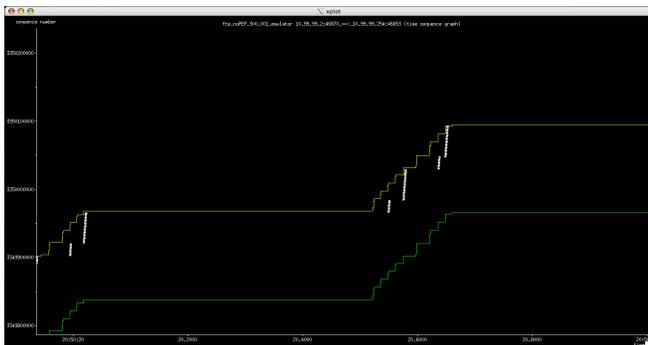


Figure 2: 1 second snippet of FTP w/o a PEP

Figure 2 shows a 1 second snippet of the TCP sequence graph for a 100 MB FTP transfer over a high latency (600 ms RTT delay) and 0.001% BER network without a PEP. Note the white marks at the beginning 20:50:20:000 timestamp have reached the maximum receive window and had to wait to resume until the receive window increases due to received ACKs at 20:50:20:600. Thus, the receive window flow control is throttling back the transmissions and the 600 ms RTT delay is slowing recovery time. The result is that our FTP application (NcFTP 3.2.2) has a maximum throughput of approximately 730 kbps at 600 ms delay regardless of BER.

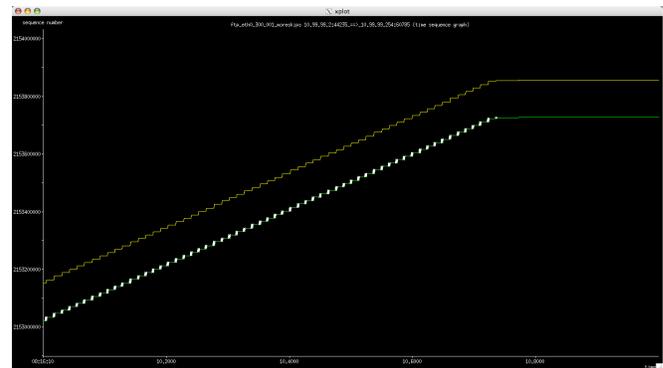


Figure 3: 1 second snippet of FTP transfer captured between the client and the PEP

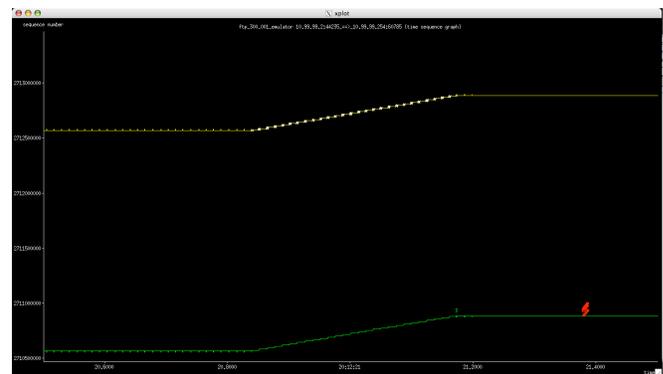


Figure 4: 1 second snippet of FTP transfer captured on the satellite emulator after the PEP

Figures 3 and 4 show the TCP sequence graph for 100 MB FTP transfer using a PEP. Figure 3 is the TCP trace captured between the client and the PEP, while Figure 4 is the TCP trace captured on the Wide Area Network (WAN). Note the differences lie in the TCP window size and the presence (or absence) of retransmitted packets. Figure 3 shows that the TCP window size remains quite small as seen by the client. However, we see in Figure 4 that that the TCP window size on the WAN side of the PEP is much larger than in either Figure 2 or 3. This shows the effect of the PEP, allowing clients to maintain small TCP window sizes while maintaining efficiency on high latency WAN links by increasing the TCP window size.

Figures 5, 6, 7 and 8 show similar time sequence graphs for an SCP file transfer over the same 600 ms RTT delay, 0.001% BER network.

Figure 5 shows a 1 second snippet of an SCP file transfer without a PEP. Compared to Figure 2 we see that the TCP window size is much larger than for the FTP application.

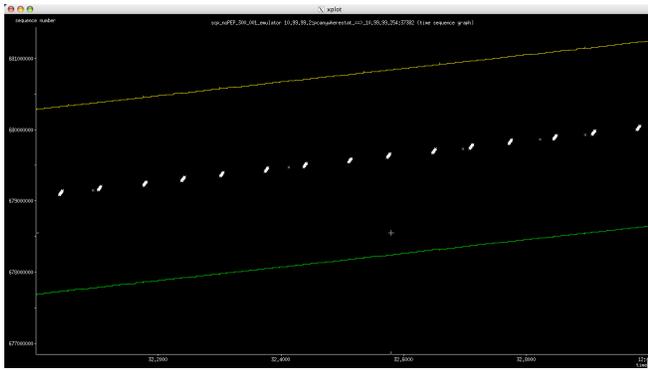


Figure 5: 1 second snippet of SCP file transfer w/o PEP

Figure 6 shows a 1 second snippet of an SCP file transfer using a PEP, captured between the client and the PEP. Here again, we see that the TCP window size between the client and the PEP remains small as with utilizing the PEP for the FTP transfer.

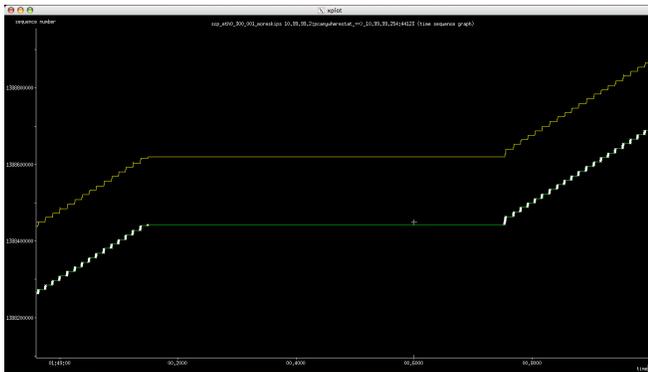


Figure 6: 1 second snippet of SCP file transfer captured between the client and the PEP

Figure 7 shows a 1 second snippet of an SCP file transfer captured on the WAN side of the PEP. Again, as with FTP we see that the WAN side shows a large TCP window. Of note, we also see a number of retransmissions and duplicate ACK packets (as indicated by the red marks and the green tick marks respectively).

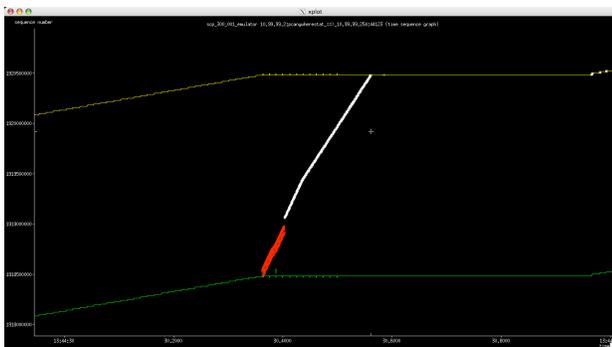


Figure 7: 1 second snippet of SCP file transfer captured on the WAN after the PEP

Figures 8 and Figure 9 illustrate two of the ways SCP increases file transfer efficiency on high latency and high BER networks. Figure 8 shows SCP increasing the TCP window size at the beginning of the file transfer. In the portion of the trace shown, the receive window size grows from approximate 220,000 bytes to 1,000,000 bytes. Figure 9 shows the use of the TCP selective acknowledgments (SACK) option (denoted by the purple marks). With selective acknowledgments, the data receiver informs the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost [18].

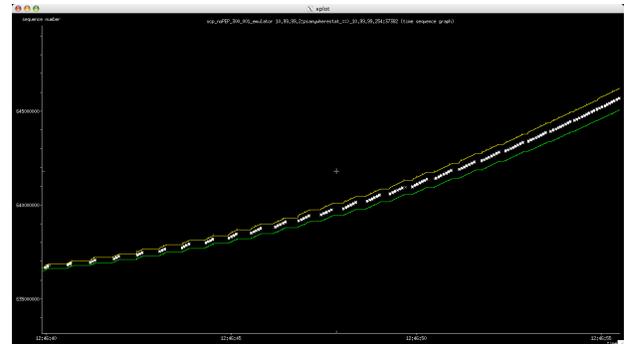


Figure 8: Increasing TCP window size for SCP file transfer w/o a PEP



Figure 9: SCP making use of the TCP SACK option

8. SUMMARY AND CONCLUSIONS

A NACK-based file transfer protocol such as Saratoga will out perform TCP-based file protocols that use modern TCP implementations or a PEP. This was the expected result

A PEP designed to improve TCP performance over large BDP link will not improve interactive communications of single packet transfers. This also was the expected result.

For our particular system, a high BDP link with no competing traffic and very few errors, the self-tuning capabilities of modern TCP implementation provide nearly identical performance to deployment of a PEP and require no configuration or tuning. The SCPS PEP (and most if not

all other PEPs) must be configured for the BDP characteristics of the link they are compensating for. If the link BDP changes, the PEP configuration must also be updated.

9. THE WAY FORWARD

The mode of operations continues to evolve with each science mission relative to the command, control and data distribution of the Global Hawk science payload. The Principal Investigators are becoming more comfortable with controlling their specific science instruments while perhaps turning control of data retrieval over to the overall payload control personal. New concepts are emerging whereby the person with overall responsibility for science payload control would transfer the large data sets from UAV to ground. The principal investigators and their teams can then access that data from anywhere using standard Internet protocols.

REFERENCES

- [1] Dawkins, G. Montenegro, M. Kojo, V. Magret, Vaidya: "End-to-end Performance Implications of Links with Errors," RFC 3155, BCP 50, August 2001
- [2] P. Sarolahti and A. Kuznetsov. "Congestion Control in Linux TCP". In Proceedings of Usenix 2002. Monterey, CA, USA, June 2002.
<http://www.sarolahti.fi/pasi/papers/linuxtcp.pdf>
- [3] S. Ha, I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," ACM SIGOPS Operating System Review, Volume 42, Issue 5, July 2008, Page(s): 64-74, 2008
- [4] Kun Tan, Jingmin Song, Qian Zhang, and Murari Sridharan: "A Compound TCP Approach for High-speed and Long Distance Networks," July 2005, <http://research.microsoft.com/apps/pubs/default.aspx?id=7018>
- [5] L. Wood, W. Eddy, C. Smith, W. Ivancic, C. Jackson: "Saratoga: A Scalable File Transfer Protocol," draft-wood-tsvwg-saratoga-10, September 16, 2011 (Work in Progress)
- [6] W. Eddy, L. Wood, W. Ivancic: "TFRC-based Congestion Control for Saratoga," draft-eddy-tsvwg-saratoga-tfrc-00.txt, September 26, 2011 (Work in Progress)
- [7] "CCSDS Recommended Standard For SCPS Transport Protocol (SCPS-TP)," CCSDS 714.0-B-2, October 2006 <http://public.ccsds.org/publications/archive/714x0b2.pdf>
- [8] Lisong Xu, Khaled Harfoush, and Injong Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks", Proceedings of IEEE INFOCOM 2004, pp. 2514-2524, HongKong, March, 2004.
- [9] <http://channel-emulator.grc.nasa.gov/>, October 2011
- [10] Sangtae Ha, Injong Rhee and Lisong Xu, CUBIC: A New TCP-Friendly High-Speed TCP Variant, ACM SIGOPS Operating System Review, Volume 42, Issue 5, July 2008, Page(s):64-74, 2008.
- [11] <http://linux-ip.net/articles/Traffic-Control-HOWTO/>, October 2011
- [12] <http://www.linuxfoundation.org/collaborate/workgroups/networking/netem>, October 2011
- [13] <http://www.nuttcp.net/>, October 2011
- [14] Space Communication Protocol Standards (SCPS), <http://www.openchannelsoftware.com/projects/SCPS>, October 2011
- [15] <http://www.tcptrace.org/index.html>, October 2011
- [16] <http://www.tcpdump.org/>, October 2011
- [17] <http://www.xplot.org/>, October 2011
- [18] M. Mathis, J. Mahdavi, S. Floyd TCP: "Selective Acknowledgment Options," RFC 2018, October 1996

Biographies



Patrick Finch is a research scientist employed by the University Corporation at Monterey Bay working for NASA at Ames Research Center. Most of his work revolves around building hardware and software command and control solutions for remotely operated instruments. He also designs web service interfaces to NASA resources allowing better data dissemination and has interests in real-time data collection and dissemination for disaster response.



Don Sullivan works for NASA at the Ames Research Center, in the Silicon Valley, California. Currently, his main interest is in network communication protocol design, both terrestrial and satellite based, enabling the implementation of ad-hoc sensor webs. He has designed the communication subsystems and ground and space based data dissemination systems for more than eight NASA UAVs, starting with the Hawai'i based, solar powered, Pathfinder in the 1990s.



William Ivancic has over twenty-five years of experience in network and system engineering for communication applications, communication networking research, state-of-the-art digital, analog and RF hardware design and testing. He currently is a senior research engineer at NASA's Glenn Research Center where he directs the hybrid satellite/terrestrial networking, space-based Internet, and aeronautical Internet research. He has lead research efforts to deploy commercial-off-the-shelf (COTS) technology into NASA missions including the International Space Station and Shuttle. Mr. Ivancic has recently performing research on advance routing research for space-based and aeronautic-based networks. Of particular interest is large scale, secure deployment of mobile networks including mobile-ip and mobile router technology.

Mr. Ivancic is also principle of Syzygy Engineering, a small consulting company specializing in communications systems and networking as well as advanced technology risk assessment. Mr. Ivancic is currently performing research and development on Identity-based security and key and policy management and distribution for tactical networks - particularly mobile networks.