ON THE PERFORMANCE OF TCP SPOOFING IN SATELLITE NETWORKS

Joseph Ishac
NASA Glenn Research Center
Satellite Networks and Architectures Branch
Cleveland, Ohio
jishac@grc.nasa.gov

Mark Allman
BBN / NASA Glenn Research Center
Satellite Networks and Architectures Branch
Cleveland, Ohio
mallman@bbn.com

ABSTRACT

In this paper, we analyze the performance of TCP in a network that consists of both satellite and terrestrial components. One method, proposed by outside research, to improve the performance of data transfers over satellites is to use a performance enhancing proxy often dubbed "spoofing". Spoofing involves the transparent splitting of a TCP connection between the source and destination by some entity within the network path. In order to analyze the impact of spoofing, we constructed a simulation suite based around the network simulator ns-2. The simulation reflects a host with a satellite connection to the Internet and allows the option to spoof connections just prior to the satellite. The methodology used in our simulation allows us to analyze spoofing over a large range of file sizes and under various congested conditions, while prior work on this topic has primarily focused on bulk transfers with no congestion. As a result of these simulations, we find that the performance of spoofing is dependent upon a number of conditions.

1 INTRODUCTION

A growing area of interest is that of hybrid networks, or networks that contain both terrestrial and wireless links. While there are many forms of hybrid networks, the work presented in this paper focuses on the use of a geosynchronous satellite within a network path. More specifically, the satellite link is located just prior to the user, as is found in a number of real-world situations. For instance, satellite companies are providing Internet services to consumers using direct-broadcast satellites. In addition, NASA is interested in delivering data collected by its space assets to investigators at various locations around the world via satellite. Finally, satellite transmission is a good match for communication between military troops in the field and military commanders. One of the main disadvantages of using a GEO satellite in network communication that has been outlined in previous literature is that current internetworking protocols do not quickly adapt to the available bandwidth when traversing a network with a long delay.

The Transmission Control Protocol (TCP) [RFC793] is the most widely used transport protocol for Internet traffic. One TCP feature in particular, congestion control, incorporates the slow-start algorithm, which may cause performance degradation on high delay links [AKO00]. The end result is a decrease in initial performance, since it takes longer to build up the sending rate over a long-delay network path than a short-delay path. Spoofing, which is discussed in more detail in Section 2, has been proposed mainly for solving the problem with TCP's startup speed over networks with high delays. However, prior work on spoofing has focused on simulations of bulk transfers without competing traffic, and has thus left an incomplete picture of spoofing's overall performance. The project discussed in this paper focuses on creating a simple yet versatile simulation environment, in which the performance of spoofing can be assessed across a large range of file sizes, while under congested conditions.

We also note that this paper only presents investigations into the performance of TCP spoofing. While the performance of such a mechanism is an important point when deciding whether or not to use a spoofer, it is not the only consideration. We encourage network operators to read [RFC3135] to gain a better understanding of some of the non-performance based considerations involved in deciding whether or not to use a spoofer.

The remainder of the paper includes Section 2, which discusses spoofing in greater depth, and Section 3, which outlines the actual simulation mechanics and details. Section 4 presents the results of those simulations. Finally, Section 5 summarizes the conclusions and lists possible areas for future work on this subject.

2 BACKGROUND

In an attempt to mitigate the disadvantages of TCP over long-latency links, researchers have been introducing performance-enhancing proxies (PEPs) into networks. One such PEP that is currently being used in satellite networks is TCP spoofing [RFC3135, BB95, ASBD96]. The objective of spoofing involves isolating the long-latency link by introducing a middle agent which splits the TCP connection (see Figure 1). However, unlike a proxy cache, spoofing is transparent to both the sender and receiver. Thus, the middle agent, or "spoofer", takes on the personality of both parties. The responsibility of the spoofer is to inter-

cept, cache, and acknowledge data received by the sender and then forward that data to the receiver. As a result, spoofing breaks the end-to-end semantic of TCP. While this raises several philosophical issues [RFC3135], those issues are not the focus of this paper. Finally, it is worth noting that in our model data segments and connection teardowns are spoofed, while connection setup remains end-to-end.

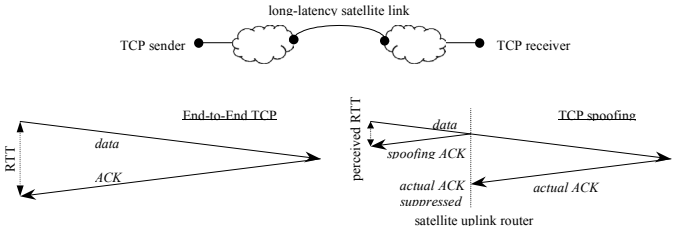

Figure 1: Satellite Spoofing

3 SIMULATION OVERVIEW

Terminology

For our simulations we used two metrics to measure the performance of a network flow. The first, throughput, is a measure of the time needed to transfer a certain amount of data and can be measured from either the sender's or receiver's perspective. For a sender side analysis, the time of completion is marked by the reception of the ACK for the final data packet, whereas the time of completion for receiver side analysis is marked upon transmission of the final ACK. The second metric used in the simulation was a measure of the number of dropped data packets.

Topology

The test network consists of five hosts and five routers as shown in Figure 2. Each host is connected to its appropriate router via an Ethernet link and runs TCP with selective acknowledgements (SACK) [RFC2018, FF96] and delayed ACKs [RFC1122, RFC2581]. Routers enforce drop-tail queuing on all links. Finally, segment sizes of 1500 bytes were used, as [All00] shows are common for bulk transfers on the Internet.

The topology is laid out such that there are distinct satellite and Internet portions. Delay over the satellite is fixed at 250ms, and the download and upload capacity set at a T1 rate. While the upload rate is much higher than what is economically feasible for a home user, the effect of variances to the transmission capacity was not of interest for this set of experiments. The second major portion of the topology, the Internet model, consists of four nodes. The two hosts in the model are responsible for generating cross traffic over the Internet link. The link between the two routers acts as the "Internet", whose bandwidth and delay are set to 1.5 Mbits per second and 69 milliseconds

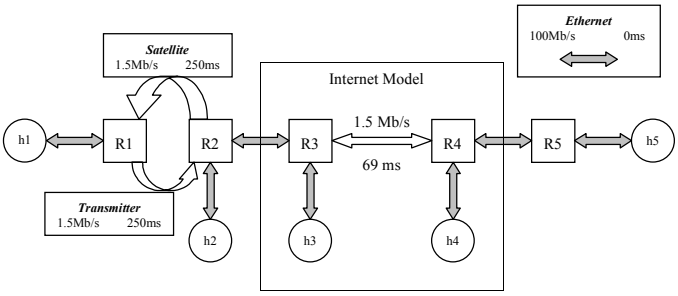

Figure 2: Network Topology

Queue Size
Ethernet = ∞
Others = (Bandwidth · (2·Delay)) / Segment Size
(Only the integer part of the result is taken from all calculations)

Equation Set 1: Queue Calculations

respectively. While the Internet has a physically larger capacity than the one used, the allocation of bandwidth is regulated and so an actual attainable value is less. The delay used was calculated by sampling the delay of several sites at various geographical distances and averaging the result. Also, changes to the capacity and delay of the Internet link result in predictable effects on the measured metrics. Thus, the bandwidth and delay used are sufficient in characterizing a typical Internet path. The results of the varying bandwidth and delay tests were left out due to space considerations. Even though the model for the Internet is simple and unrealistic, it is sufficient in capturing the basic characteristics of propagation delay, limiting bandwidth, and competing traffic.

The remaining hosts are the nodes at which connections of interest will take place. Using these three hosts, any combination of the hybrid network can be analyzed. More specifically, *h1* represents a pure satellite user, and *h5* represents a server with a high-speed connection to the Internet. Host *h2* is at a middle ground with access to both portions of the network. This property also makes *h2* capable of spoofing connections as shown in Figure 3.

Thus, with spoofing enabled, a connection from *h5* to *h1* would be spoofed at *h2*. It would cache, with infinite capacity, data received from *h5* and forward the data to *h1*.

Traffic

All transfers used in the simulation make use of the File Transfer Protocol (FTP) to transmit data. No competing traffic is present on the satellite link as it represents a dedicated satellite channel.[1] However, competing traffic is

[1] The satellite in this simulation was based off of the Advanced Communications Technology Satellite (ACTS), which supported

present on the Internet link by using an analytical FTP generator which is based on models given in [Pax94] and discussed in detail in [Ish01]. Depending on the simulation, the desired number of competing flows is started in alternating fashion between *h3* and *h4*. Start times are based on a random Poisson generated mean, which is included in the traffic profile. In order to keep the traffic load at a desired level, new competing flows are initiated to replace ones that should expire. The new flows are initiated using the same random mean in order to avoid synchronization and phase effects. Finally, the main flow is started five seconds into the simulation in order to avoid the startup effects of the competing flows.

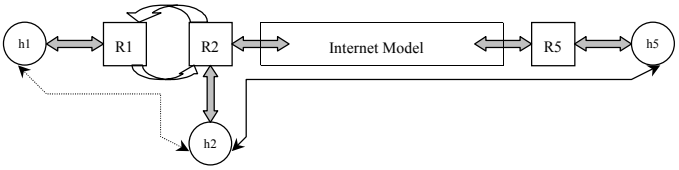

Figure 3: Spoofing in the Simulator [2]

Software

The simulations in this paper make use of the *Network Simulator* (ns) [NS] version 2.1b8. Generation of the traffic profile is separated from the simulation so as to facilitate reuse and modularization. The overall layout of software structure is shown in Figure 4. The output from ns consists of three trace files which are uniquely named in relation to the type of simulation being done. This allows the analyzer to distinguish which traces to analyze and also allows for congruent execution of simulations. The function of the controller is to synchronize the spawning and execution of both programs. Finally, a script automates the entire process, synchronizing traffic generation with the controller and allowing for multiple runs of different case scenarios.

4 RESULTS

The results detailed in this section are based on a 30 run simulation with the following characteristics:

- Transmission of files from the network user (*h5*) to the satellite user (*h1*) under the following granularity: 0 to 100 packets by 1 packet, 110 to 500 packets by 10 packets, 600 to 900 packets by 100 packets, 1000 to 2000 packets by 1000 packets.
- Analysis of both end-to-end TCP connections and spoofing as well as sender and receiver side analysis.
- A congestion level of 50 competing flows.

For the sake of simplicity, end-to-end TCP is referred to as "Regular" TCP in any subsequent plots.

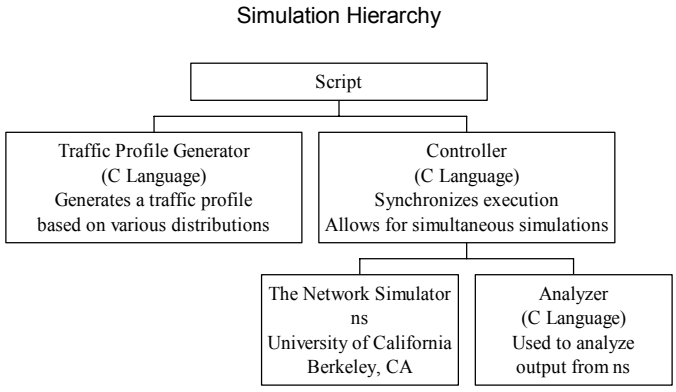

Figure 4: Software Layout

Figure 5 shows the average throughput of the main flow (*h5*→*h1*) from both the receiver and sender side. This in combination with spoofing represents the four curves on the plot. For long transfers, the throughput values are nearly the same for Regular TCP regardless of whose viewpoint is taken (as one would expect since the sender and receiver are only separated by a short amount of time). Also, the same observation holds for Spoofed TCP.

Figure 6 shows the percent difference in throughput as reported in Figure 5. The percent difference is calculated by taking the throughput difference of spoofing and end-to-end TCP over the throughput of end-to-end TCP. Thus, the line identified as "Sender Side" in the plot refers to the percent difference between "Regular-Sender" and "Spoofing-Sender". A positive percent difference indicates that the PEP outperformed the base condition, while a negative value indicates the exact opposite.

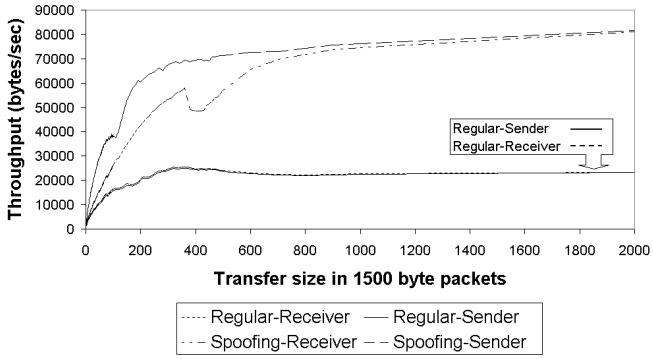

Figure 5: Throughput vs. Transfer Size

packet switching, spot transmissions, and frequency reuse. ACTS also used forward error correction to eliminate nearly all corruption at T1 speeds. Thus, our modeled network does not corrupt packets.

[2] Due to simulator restrictions, spoofing could not be done at routers, as would likely be the case if it were implemented in a real network. However, moving it to the host adds only the Ethernet delay which is negligible and likely much smaller than any processing delays that would be present in a real system.

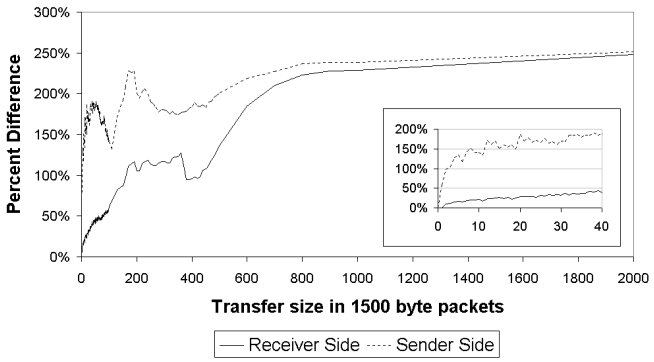

Figure 6: Percent Difference of Throughput

Also included in the percent difference plot is an enlarged view of transfers consisting of 40 packets or less. The importance of considering small transfers is shown in Figure 7.[3] From this plot, we see that transfers consisting of ten packets or less account for 90% of all connections. Although the data represents only a single network, the underlining concept has been generally noted in other networks as well.

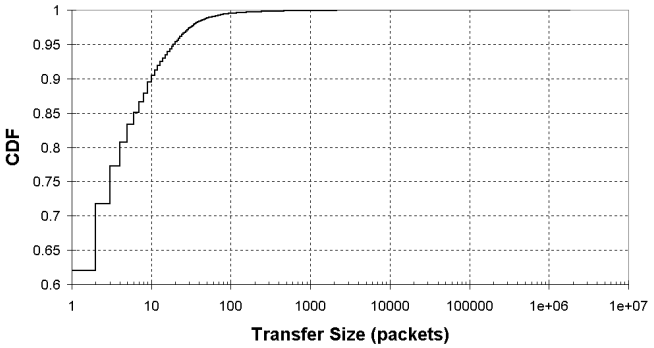

Figure 7: Distribution of Transfer Sizes

From Figure 6 we see that spoofing has different effects depending on which vantage point is used. From the receiver's viewpoint, there is little effect for small transfers - at most a 20% gain for transfers of ten packets or less. However, from the sender's perspective, the gain is much larger for those same small transfers. The large gain in throughput for the senders could be beneficial for busy servers. By freeing resources associated with long-delay connections more rapidly, spoofing may allow servers to satisfy more requests. For large transfers, both viewpoints show approximately the same performance gain. Also, the percent difference graph contains some fluctuation, which refers to the several short and successive increases and declines within the plot. Such fluctuations can be attributed to the fact that the throughput values are an average of 30 runs and that the variance is large. However, the dip in throughput found around 400 packets in Figure 6 is rather unusual.

To better understand why this dip occurs, we turn to the number of drops. A plot for the number of dropped data segments is shown in Figure 8. This plot follows the general principle of that seen with throughput, although dropped packets are measured along the entire network path. Figure 8 shows a sharp increase around 400 packets when spoofing is used. Thus, one possible cause of the decrease in throughput is that the spoofer receives data more quickly, and thus overruns the satellite channel, dropping a large number of packets. This is possible since spoofing allows the rate of incoming packets to increase over the Internet and accumulate at the spoofer, which is still in early slow-start. Thus, spoofing adds a second bottleneck into the network path.

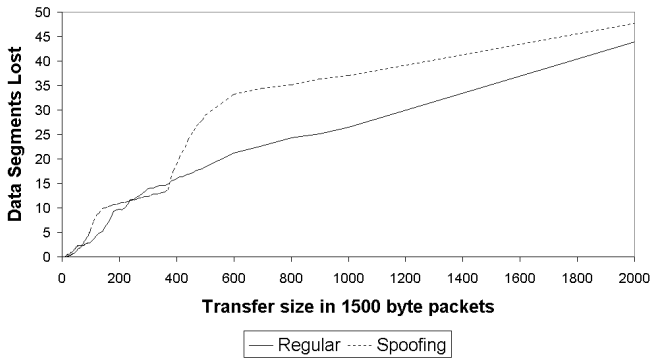

Figure 8: Data Loss vs. Transfer Size

We next turn our attention to how varying the amount of congestion on the Internet portion of the network path effects TCP spoofing performance. We varied the number of FTP sessions being conducted across the Internet path from one to 200 - or from an approximate drop rate of 0.2% to roughly a drop rate of 9%. Figure 10 shows the percent difference in throughput, from the sender's viewpoint, between Regular and Spoofed TCP as a function of transfer size for various levels of cross traffic. As in all other plots in this paper, each point is the mean of 30 simulations. From the figure, there does not appear to be a clear pattern to the gain spoofing provides to the sender in relation to the level of congestion. However, we note that the gain is significant in all cases - at least 150% for transfers of at least 15 packets.

Figure 9 shows the same results as illustrated in Figure 10, except from the receiver's vantage point. In this case, there is a significant difference in the performance seen by the user as the level of congestion in the network changes. As the number of competing flows increases the performance gain realized from using TCP spoofing also increases. This is explained by TCP's ability to recover from conges-

[3] Cumulative Distribution Function (CDF) of transfer sizes in packets as seen at the NASA GRC firewall in late 2000 (7 days of traffic yielding 8 million connections).

tion losses in the Internet more rapidly when using spoofing than when using the long-feedback end-to-end TCP. We note that small transfers (the predominant kind of transfer) still observe only modest increases in performance. This is caused by the lack of loss on these connections (since they are short).

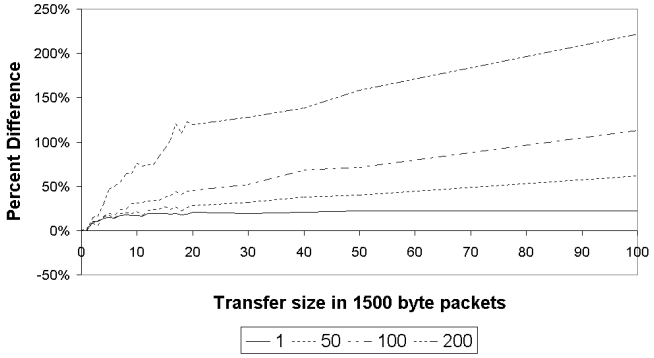

Figure 9: Effect of competing flows on receiver side throughput

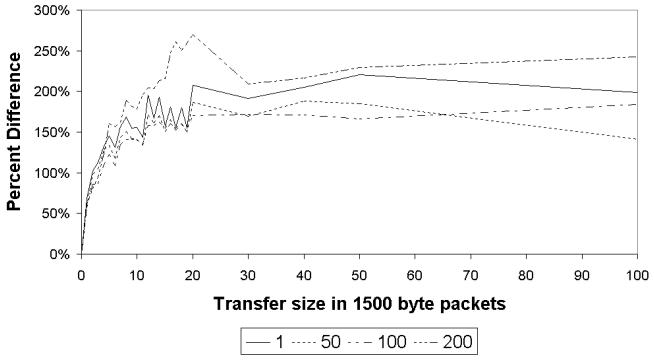

Figure 10: Effect of competing flows on sender side throughput

5 CONCLUSIONS AND FUTURE WORK

The simulations presented in this paper offer a mixed set of results on the efficacy of spoofing. Our key conclusions are:

- We found that spoofing is indeed beneficial for large file transfers (confirming previous results in the presence of network congestion).
- For small transfer sizes, spoofing increases the throughput as observed by the data sender.
- Spoofing was much less beneficial for throughput observed at the receiver, which is the vantage point perceived by the end-user. Since a majority of data sent across networks is small and sent to the user, spoofing will likely not provide a large advantage from the user's perspective.
- Spoofing's benefit to web servers and other content providers may be significant.
- Spoofing allows for data to accumulate at the spoofer, creating a second bottleneck and increasing the number of dropped data packets, which also degrades the receivers perceived performance. Future work should include attempts to mitigate this problem while still retaining the performance benefits of spoofing for larger transfers.
- The performance benefits of spoofing increase as the amount of network congestion grows.

We realize that the work done in this paper reflects simulations and not actual measurements on real networks. Thus, a natural extension of this work may involve implementing these simulations in actual network test beds. Other extensions involve the inclusion of other types of PEPs as well as the effect of changes to the asymmetry of the satellite.

Finally, we note that while this paper is only concerned with the performance implications of TCP spoofing there are several architectural issues that need to be considered when deciding whether or not to use TCP spoofing. We suggest that readers look at [RFC3135] for a discussion of those topics.

6 ACKNOWLEDGEMENTS

We would like to thank Vern Paxson for the various discussions in regards to the FTP algorithms, and Funda Ergun for commenting on earlier drafts of this report.

7 REFERENCES

[AKO00] M. Allman, H. Kruse, and S. Ostermann, “A History of the Improvement of Internet Protocols Over Satellites”, ACTS Conference 2000, May 2000

[All00] M. Allman, “A Web Server’s View of the Transport Layer”, ACM Computer Communication Review, 30(5), October 2000

[ASBD96] V. Arara, N. Suphasindhu, J. Baras, and D. Dillon, “Asymmetric Internet Access Over Satellite-Terrestrial Networks”, 16th International Communications Satellite Systems Conference and Exhibit, Part1, Washington, D.C., Feb. 1996

[BB95] A. Bakre, B. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", 15th International Conference on Distributed Computing Systems, May 1995

[FF96] K. Fall and S. Floyd, “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”, Computer Communications Review, 26(3), July 1996

[Ish01] J. Ishac, “FTP Traffic Generator”, Technical Report, Case Western Reserve University, January 2001

[NS] VINT project: http://www.isi.edu/nsnam/vint/index.html

[Pax94] V. Paxson, “Empirically-Derived Analytic Models of Wide-Area TCP Connections”, IEEE/ACM Transactions on Networking, 4(2), August 1994

[RFC1122] R. Braden, “Requirements for Internet Hosts -- Communication Layers”, RFC 1122, October 1989

[RFC2018] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP Selective Acknowledgment Options”, RFC 2018, Oct. 1996

[RFC2581] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control”, RFC 2581, April 1999

[RFC3135] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, “Performance Enhancing Proxies”, RFC 3135, June 2001

[RFC793] J. Postel, “Transmission Control Protocol”, RFC 793, September 1981