EFFECT OF TRANSFER FILE SIZE ON TCP-ADaLR PERFORMANCE: A SIMULATION STUDY

Modupe Omueti and Ljiljana Trajković
Simon Fraser University
Vancouver
British Columbia
Canada
{momueti, ljilja}@cs.sfu.ca

ABSTRACT
Large file transfers have become common with the increasing demand for bandwidth-intensive Internet applications. The transmission control protocol with adaptive delay and loss response algorithm (TCP-ADaLR) was proposed to improve TCP performance in the presence of large propagation delays and error losses. In this paper, we analyze the effect of transfer file size on the TCP-ADaLR performance in networks with large propagation delays. We employ the file transport protocol (FTP) file download application to evaluate TCP-ADaLR performance for varying file sizes using. We compare the performance of TCP-ADaLR with TCP NewReno and TCP SACK. Simulation results show that TCP-ADaLR mechanism improves TCP performance in cases of large file transfers in lossless networks with large propagation delays. When the file size is larger than 50 MB, TCP-ADaLR improves TCP throughput while the throughput of the TCP NewReno and TCP SACK connections remain unchanged. Furthermore, when transfer file size is 500 MB TCP-ADaLR throughput increases up to 75 % compared to TCP NewReno and TCP SACK.

KEY WORDS
Communication networks, TCP, file transfers, propagation delays, simulation, performance evaluation.

1. Introduction

There is growing demand for new high-speed multimedia and data Internet applications [1]. These applications generate large files that are exchanged across various types of communication networks. File transfer protocol (FTP) [2], the primary protocol used to transfer files, relies on the transmission control protocol (TCP) [3] for reliable transport.

To achieve reliable data transfers, TCP employs acknowledgement (ACK) mechanisms for connection management and flow and congestion control. The ACK mechanism is adversely affected by large propagation delays, which results in large round trip times (RTTs). Geostationary earth orbit (GEO) satellite networks are characterized by such large propagation delays and by high bit error rates (BERs) [4].

We had proposed TCP with adaptive delay and loss response algorithm (TCP-ADaLR) [5], [6] for deployment in GEO satellite networks. For small to moderate files downloaded via FTP and hypertext transfer protocol (HTTP) applications, TCP-ADaLR exhibits improved performance. TCP is adequate for FTP transfers of small to moderate files in networks with small RTTs. In the absence of losses, throughput of FTP file transfer is limited by the TCP receiver's advertised window (*rwnd*) and large RTTs. For large RTTs, this throughput limitation becomes evident as file sizes increase. In this case of large RTTs, we investigate the scalability of TCP-ADaLR and its ability to accelerate file transfers. We also evaluate the performance of TCP-ADaLR in terms of FTP download response time, TCP throughput, link throughput, and link utilization. We compare its performance with TCP NewReno and TCP SACK and conclude that TCP-ADaLR is a viable TCP option for large file transfers in lossless networks with large propagation delays or high latencies.

This paper is organized as follows: An overview of TCP and the impact of window size on TCP performance is given in Section 2. In Section 3, we present background and related work. A description of the TCP-ADaLR modifications for networks with high BERs and large propagation delays is given in Section 4. Simulation scenarios and results of the performance evaluation of TCP-ADaLR and TCP NewReno are presented in Section 5. We conclude with Section 6.

2. Overview of the Transmission Control Protocol

TCP provides connection-oriented and reliable byte-stream delivery services for Internet application-level protocols such as FTP and HTTP [7]. TCP utilizes window flow control to adjust its transmission rate. The

congestion window (*cwnd*) and *rwnd* determine the volume of data transmitted through the network. At the start of transmission, the TCP sender's initial window specifies the number of segments (window) that a TCP sender may transmit without receiving an ACK from a receiver [3], [8]. When a TCP sender receives ACKs of transmitted segments, it slides (increases) the window size based on the value of *rwnd* and the number of segments or bytes acknowledged.

A large bandwidth-delay product characterizes networks with large propagation delays. The bandwidth-delay product defines the amount of data required to be unacknowledged (in–flight) in order to fully utilize the available link capacity. For a transmission link, the bandwidth-delay product is the product of the link capacity and the RTT [9]. A communication protocol should be able to transmit this amount of data through the network. Network operators often increase the available link bandwidth in order to eliminate bandwidth as a limiting factor in throughput for file transfers. However, TCP throughput depends on *rwnd* and RTT (*rwnd*/RTT). Hence, if packet losses are ignored, file transfer throughput is also limited by the maximum amount of data in flight, which depends on the maximum TCP *rwnd* value (64 KB) [3]. The achievable throughput is inversely proportional with RTT and, hence, links with larger RTTs are underutilized, as shown in Figure 1. For example, only ~ 1.3 Mb/s throughput can be achieved with 100 Mb/s link capacity and RTT of 500 ms.

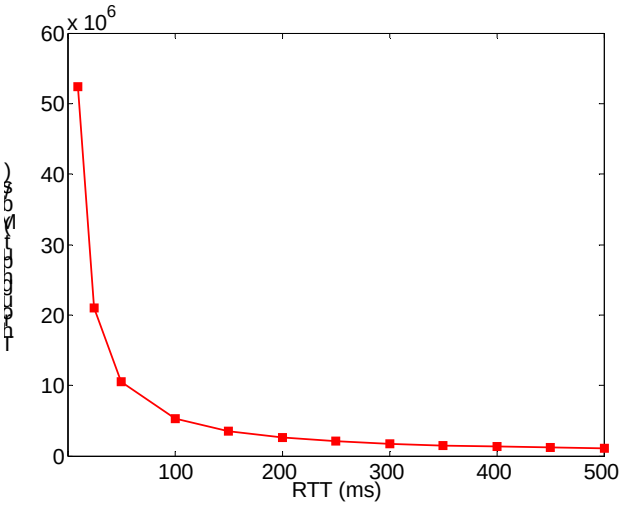

Figure 1. TCP throughput as a function of RTT. TCP throughput decreases as RTT increases.

3. Background and Related Work

HighSpeed TCP (HSTCP) [10] and scalable TCP [11] are TCP variants proposed for large bandwidth-delay product networks. HSTCP adaptively adjusts the *cwnd* based on the current *cwnd* when an ACK is received or when a segment is lost. Scalable TCP adjusts the *cwnd* by a factor $\alpha = 0.01$ upon receipt of an ACK and by a factor $\beta = 0.125$ when segment loss is detected. However, the *cwnd* adjustment factors are specifically optimized for Gigabit Ethernet wide area networks (WANs).

User Datagram Protocol (UDP) [12] has also been proposed as an alternate transport protocol for accelerating large file transfers in networks with large RTTs. However, UDP is an unreliable connectionless protocol. Reliable blast UDP (RBUDP) [13] employs TCP for exchanging control information and UDP for transferring data. However, for optimal performance, RBUDP requires a priori knowledge of the rate of the bottleneck link between the sender and receiver. For data transfers, simple available bandwidth utilization library (SABUL) [14] combines UDP with rate-based control and TCP-like window-based control.

4. TCP with Adaptive Delay and Loss Response Algorithm

We proposed TCP-ADaLR to mitigate the effect of large propagation delays, error losses, and the limited TCP *rwnd* on TCP throughput. TCP-ADaLR is an end-to-end algorithm that introduces the sender scaling component ρ, adaptive window (*cwnd* and *rwnd*) increase, and loss recovery mechanisms at the TCP sender [5], [6]. The adaptive window increase mechanisms enable a TCP sender to better utilize the available link bandwidth in the absence of losses. These mechanisms adjust the window increments by the scaling component ρ, which is computed based on RTT measured from acknowledged TCP segments. The loss recovery mechanism allows a TCP sender to recover more quickly from error losses than in case of TCP NewReno or TCP SACK .The TCP-ADaLR modifications have been applied to TCP NewReno and TCP SACK. These modifications are well suited for networks with large propagation delays and high BERs. Performance evaluation of TCP-ADaLR for small file transfers (HTTP application) and moderate file transfers (FTP file download application) indicates that TCP-ADaLR improves TCP performance [5], [6]. However, for large file sizes, further performance improvement would be required for better throughput and link bandwidth utilization.

5. Simulation Scenarios and Performance Evaluation

We simulate a heterogeneous network consisting of wired and wireless segments. The network is asymmetric with distinct data rates of the forward and reverse transmission paths between the gateway and the client shown in Figure 2. A 10 Mb/s full-duplex Ethernet link connects the server and the gateway. All shown propagation delays are

measured one-way and the links are lossless. TCP simulation parameters are shown in Table 1.

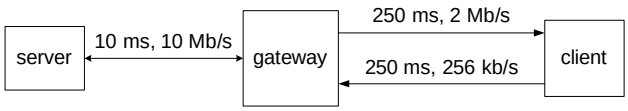

Figure 2. Topology of a heterogeneous network.

Table 1. TCP simulation parameters.

TCP Parameters	Value
Sender maximum segment size (SMSS)	1,460 bytes
Slow start initial count	2 SMSS
Receiver's advertised window	65,535 bytes
Timer granularity	0.5 s
Persist time-out	1.0 s
Maximum ACK delay	0.0 s
Maximum ACK segment	1
Duplicate ACK threshold	3
Retransmission threshold	6
Initial RTO	3.0 s
Minimum RTO	1.0 s
Maximum RTO	64. 0 s
RTT gain	0.125
RTT deviation coefficient	4
Deviation gain	0.25

We evaluate the performance of TCP-ADaLR using the OPNET discrete event network simulator [15]. OPNET simulation variables are defined in the *simulation set info* menu. The variable *duration* is the specified simulated real-time. The variable *seed* is the random generator seed used for the simulation run. Multiple seeds may be defined using the *multiple seed values* attribute. The variable *values per statistic* is the interval between consecutively collected simulation results. For example, for the duration equal to 18,000 s and *values per statistic* set to 3,600, the interval between consecutively collected simulation results is 5 s. The *update interval* indicates the number of events that are required before OPNET generates simulation updates. (A larger value of *update interval* implies a longer interval update period.) The *update interval* may be set to a large value to reduce the number of updates and, thus, reduce the duration of a simulation session. The OPNET default value for the *update interval* is 500,000 events. For all performed simulations, we set this variable to 10^6 events. OPNET simulation time is the real-time required for a simulation to complete. An OPNET simulation may terminate before the time specified by the variable *duration* elapses. For example, this occurs if the simulated FTP download application is completed.

We first simulate a 50 MB file download for various choices of the link propagation delays. The simulation parameters of the FTP file download application are shown in Table 2. We measure the time elapsed between sending an FTP request to an FTP server and receiving the complete response. This time includes the signalling delay for the connection establishment and termination. This FTP download response time indicates the user-perceived latency of the FTP file download. For TCP-ADaLR, TCP NewReno, and TCP SACK, a noticeable increase is observed in the download response time when propagation delay exceeds 200 ms, as shown in Figure 3. Larger propagation delays result in increased segment RTTs, which increases the download response time. TCP-ADaLR, TCP NewReno, and TCP SACK exhibit comparable performance for RTT less than 200 ms. When RTT exceeds 200 ms, TCP-ADaLR exhibits 5.5%–75% shorter download response time than TCP NewReno and TCP SACK. For these large RTTs, the scaling component is computed as a function of RTT and then used to adaptively adjust the window increments. Hence, TCP-ADaLR completes the file transfers faster than TCP NewReno.

Table 2. FTP file download application parameters.

Attribute	Value
File inter-request time (s)	18,000
File inter-request time distribution	constant
File size (MB)	50
File size distribution	constant
Simulated time (hours)	5

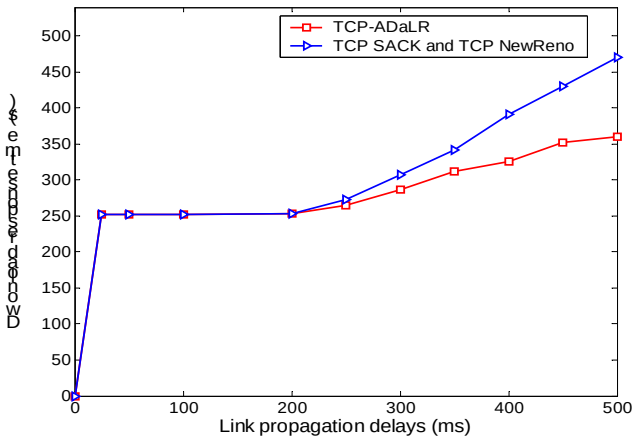

Figure 3. FTP download response time as a function of RTT for a 50 MB transfer file size. For large RTTs, TCP-ADaLR exhibits shorter download response times than TCP NewReno.

In the second simulation scenario, we vary file sizes (500 kB, 50 MB, 100 MB, 200 MB, 300 MB, 400 MB, and 500 MB). We choose the total one-way link propagation delay of 260 ms and investigate the effect on TCP-ADaLR. The

remaining file download parameters are shown in Table 2. We observe the download response time, TCP throughput, link throughput, and link utilization for TCP-ADaLR, TCP NewReno, and TCP SACK. In all shown simulation results TCP-ADaLR refers to implementations with NewReno or SACK. Both exhibit identical performance dues to lossless links. TCP NewReno and TCP SACK also exhibit identical performance.

TCP throughput is the traffic transmitted by the TCP sender to the TCP receiver. It is measured at the TCP receiver as the average bytes per second forwarded by the TCP sender and received by the TCP receiver. Link throughput, measured in b/s, is the average number of bits correctly received by the transmission link in the forward path for the duration of the file transfer. Link utilization is the fraction of the available link capacity consumed by the data transmission. It is expressed as the ratio of the number of bits correctly transmitted over the link per unit time and the link data rate. Link throughput and utilization are evaluated for the 2 Mb/s link located between the gateway and the client, as shown in Figure 2.

The download response times for various file sizes are shown in Figure. 4. As the file size increases, the FTP file download response time increases for both TCP-ADaLR and TCP NewReno connections. Furthermore, TCP-ADaLR shows 9%–38% shorter download response times than TCP NewReno. The TCP-ADaLR adaptive window increase mechanisms reduce the TCP *rwnd* window limitation and allow additional segments to be transmitted within each window. Hence, for each file size, the file download is completed faster compared to TCP NewReno.

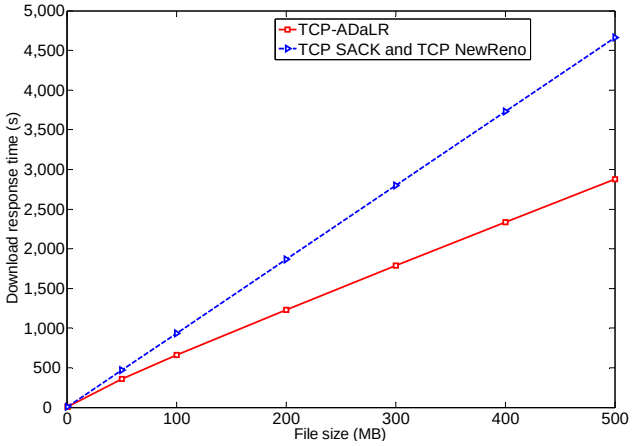

Figure 4. FTP download response time for various transfer file sizes. As the transfer file size increases, TCP-ADaLR shows shorter FTP download response times than TCP NewReno.

TCP-ADaLR algorithm improves TCP throughput when the file size increases from 50 MB to 500 MB, as shown in Figure 5. For large RTTs, TCP-ADaLR adapts its transmission rate using the scaling component. This enables TCP-ADaLR to transmit additional segments within each RTT using its adaptive *cwnd* increase mechanism. The adaptive *rwnd* increase mechanism also allows additional segments to be transmitted when the *cwnd* exceeds the *rwnd* and the unacknowledged in-flight bytes do not exceed the *rwnd*. However, the throughput of the TCP NewReno connection remains unchanged when the transfer file size increases from 50 MB to 500 MB. For large file sizes, a larger percentage of the file is downloaded during the congestion avoidance phase, which is more conservative and relies on linear *cwnd* increments. The *rwnd* limits the maximum amount of data that may be transmitted when the *cwnd* exceeds the *rwnd*. Hence, large RTTs further prevent TCP throughput increase.

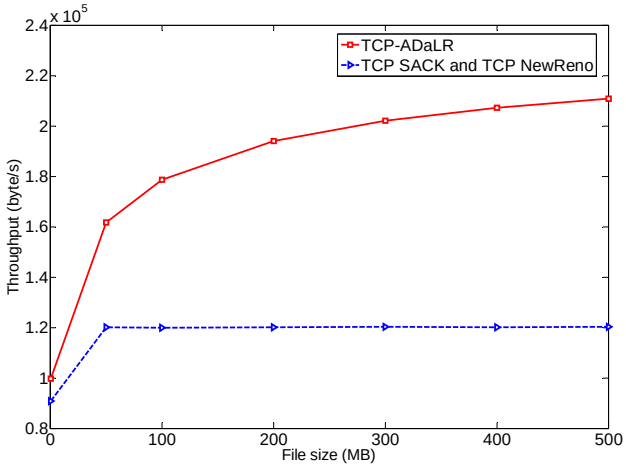

Figure 5. As the transfer file size increases, TCP-ADaLR shows 9%–75% higher TCP throughput compared to TCP NewReno.

The throughput and utilization of the 2 Mb/s link connecting the gateway and the client are shown in Figure 6 and Figure 7, respectively. As expected, the link utilization is lowest for TCP-ADaLR and TCP NewReno when the transfer file size is the smallest (500 kB). TCP-ADaLR increases link throughput and utilization for all file sizes up to 400 MB. The increasing TCP throughput enables transmission of additional segments, thus, increasing link throughput and utilization. However, the link throughput and utilization of the TCP NewReno connections decrease very slightly when the transfer file size is larger than 50 MB because of the slower transmission rate during TCP congestion avoidance phase. The value of *rwnd* limits the throughput and utilization to ~ 50% of the available capacity (2 Mb/s). However, TCP-ADaLR is able to better utilize the link capacity for larger file sizes because the window increase mechanisms adapt the transmission rate based on the measured RTTs. TCP-ADaLR shows 57%–90% higher link throughput than TCP NewReno. Hence, TCP-ADaLR performance scales with increasing transfer file sizes.

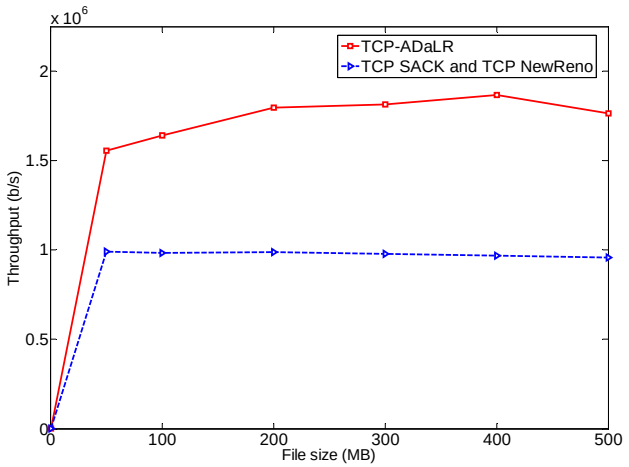

Figure 6. Link throughput as a function of transfer file size. TCP-ADaLR exhibits up to 81% higher link throughput than TCP NewReno.

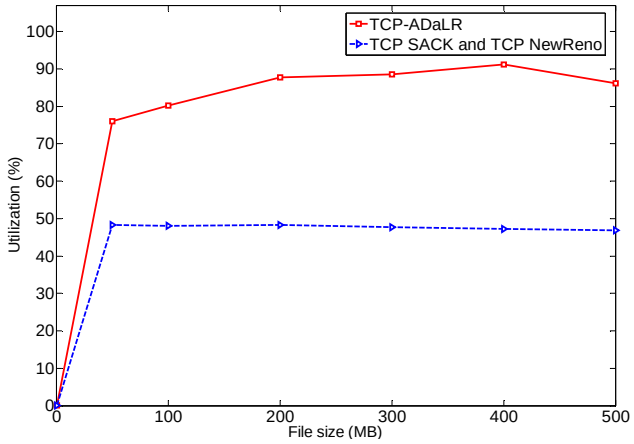

Figure 7. Link utilization as a function of transfer file size. TCP-ADaLR shows 57%–90% higher link utilization than TCP NewReno for the 2 Mb/s link.

6. Conclusion

In this paper, we simulated the effect of transfer file size on TCP-ADaLR performance and concluded that TCP-ADaLR algorithm improved TCP performance for transfers of small to moderate and large files in high latency lossless networks. Simulation results showed that TCP-ADaLR utilized link capacities better than TCP NewReno and resulted in increased TCP throughput. TCP-ADaLR also accelerated file transfers and exhibited shorter FTP download response time than TCP NewReno. The deployment of TCP-ADaLR in existing networks requires modifications only at the TCP sender. These modifications introduce additional, albeit minimal, processing and memory overheads at the TCP sender. These simulation results indicate that TCP-ADaLR is a viable option for large file transfers in existing networks that employ TCP as the transport layer protocol.

Acknowledgements

The authors would like to thank S. Lau, R. Narayanan, B. Vujičić, S. Vujičić, and W. Zeng from the Communication Networks Laboratory at Simon Fraser University for constructive comments and suggestions.

References

[1] A. Jamalipour, M. Marchese, H. Cruickshank, J. Neal, and S. Verma, "Broadband IP networks via satellites-part II," *IEEE J. Select. Areas Commun.*, vol. 22, no. 3, pp. 433–437, Apr. 2004.

[2] J. Postel and J. Reynolds, "File transfer protocol," *RFC 959*, Oct. 1985.

[3] J. Postel, Ed., "Transmission control protocol," *RFC 793*, Sept. 1981.

[4] Y. Shang and M. Hadjitheodosiou, "TCP splitting protocol for broadband and aeronautical satellite network," in *Proc. 23rd IEEE Digital Avionics Syst. Conf.*, Salt Lake City, UT, Oct. 2004, vol. 2, pp. 11.C.3-1–11.C.3-9.

[5] M. Omueti and Lj. Trajković, "TCP with adaptive delay and loss response for heterogeneous networks," in *Proc. Third Annual International Wireless Internet Conference* (*WICON 2007*), Austin, TX, Oct. 2007.

[6] M. Omueti and Lj. Trajković, "OPNET model of TCP with adaptive delay and loss response for broadband GEO satellite networks," *OPNETWORK 2007*, Washington, DC, Aug. 2007.

[7] W. Stevens, *TCP Illustrated Volume 1: The Protocols*. Reading, MA: Addison-Wesley, 1994.

[8] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *RFC 2581*, Apr. 1999.

[9] J. Sing and B. Soh, "TCP performance over geostationary satellite links: problems and solutions," in *Proc. 12th IEEE Int. Conf. on Netw.*, Guadeloupe, French Caribbean, Nov. 2004, vol. 1, pp. 14–18.

[10] S. Floyd, "HighSpeed TCP for large congestion windows," *RFC 3649*, Dec. 2003.

[11] T. Kelly, "Scalable TCP: improving performance in high-speed wide area networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, Apr. 2003.

[12] J. Postel., "User datagram protocol," *RFC 793*, Aug. 1980.

[13] E. He, J. Leigh, O. Yu, and T. DeFanti, "Reliable blast UDP: predictable high performance bulk data transfer," in *Proc. 5th Int. Conf. on Cluster Computing*, Chicago, IL, Sept. 2002, pp. 317–324.

[14] Y. Gu and R. Grossman, "SABUL: A transport protocol for grid computing," *J. of Grid Computing*, vol. 1, no. 4, pp. 377–386, Dec. 2003.

[15] OPNET Modeler software [Online]. Available: http://www.opnet.com/products/ modeler/home.html.