Experimental and Theoretical Analysis of Storage Friendly TCP Performance in Distributed Storage Area Network

Suresh Muknahallipatna, Gayathri Sivasankaran, Joseph Miles, Timothy Brothers and Nagapramod Mandagere
Department of Electrical and Computer Engineering, University of Wyoming, Laramie, WY – 82071, USA

Joseph L. White and Howard Johnson
Brocade Communications Systems. Inc., San Jose, CA – 95110, USA

ABSTRACT

Fibre channel storage area networks (SAN) are widely implemented in production data center environments. Recently the storage industry has moved towards deployment of distributed SANs (DSAN), geographically dispersed across large physical distances. In a DSAN, specialized gateway devices interconnect the individual Fibre Channel (FC) fabrics over IP networks using TCP/IP based protocols (iFCP or FCIP) or over metro to long distance optical networks such as Dense Wavelength Division Multiplexing (DWDM) based networks that utilize native FC ports supporting large numbers of link credits. When using TCP/IP based storage networking protocols to interconnect local FC fabrics in a DSAN, the sustained throughput achievable depends upon the link characteristics and TCP/IP stack implementation. Sustaining maximum possible storage traffic throughput across the wide area network enables practical DSAN deployments by maintaining the required site to site service level agreements.

This study explores the effects of several TCP/IP modifications on sustained traffic throughput for a DSAN interconnected via iFCP gateways across an impaired network. The TCP/IP stack modifications, known as storage friendly, include changes to the window scaling, congestion avoidance, and fast recovery algorithms. The theoretical background and experimental results are presented to explain and illustrate these modifications.

Keywords: TCP, Congestion, Packet Loss, Storage, Fibre Channel

1. INTRODUCTION

Fibre Channel Storage Area Networks (FC SANs) have a maximum link length of about 100 miles because of the limitations of the optical signaling electronics. Long distances require use of expensive long wavelength lasers and single mode fibers. Most FC SANs use less expensive short wavelength lasers and have maximum link length of 100 to 300 meters. Greater distances are achievable through the use of repeaters or alternative signaling techniques such as wave division multiplexing. The use of repeaters to extend range has not been widely accepted by the industry due a number of reasons [1]. Dense Wavelength Division Multiplexing (DWDM) technology over dedicated fibre optic link(s) is often deployed to directly extend FC SANs across metropolitan distances using FC equipment with sufficient buffer-to-buffer (BB) credits to maintain transfer rate. The DWDM technology over long distance (across continental USA) tends to be economically infeasible due to cost of dedicated fibre optic link(s).

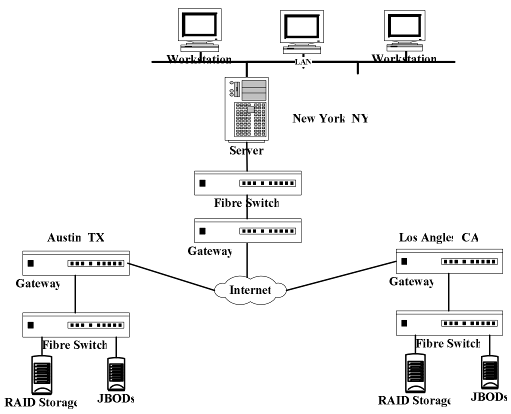

Figure 1. Distributed storage area network

The new approach to extend the range of FC SANs economically is by using non-dedicated links between servers and storages with the non-dedicated links constructed using existing internet infrastructure with FC-to-IP gateways leading to distributed SAN (DSAN). An IP based solution has the advantage of running over a wide variety of network infrastructures though commodity networking equipment. DSANs optimize and ease deployment for a variety of extended data center functions including data replication, business continuance, centralized storage access, remote device sharing, and distribution of storage resources. Figure 1 shows an example of a DSAN deployment in which three local SAN's spread across continental distances are connected across the internet via gateway devices.

A server connected to the client through a local area network is shown in New York. The server may host enterprise applications like Microsoft Exchange server, SQL server, and Oracle Database which require large storage access. The large storage at Los Angles could be used by the server on real time basis where as the storage at Austin could be for replication/backup [2]. The gateway device attached to each of these SANs allows its local devices to communicate with remote devices from the other SANs. The gateway acts as a proxy on the local SAN for the remote devices. It converts the FC protocol traffic from the device into IP storage protocol traffic. The IP traffic is sent across the internet to the remote gateway where it is converted back into FC. The storage industry has created several TCP/IP based protocols to take advantage of IP infrastructure. These are iSCSI, iFCP, and FCIP. The iSCSI protocol primarily provides direct device interconnect, while iFCP and FCIP primarily provide distance connections to existing FC devices and FC SANs.

FCIP is a tunneling protocol which connects one or more SAN islands and integrates them into a single SAN entity [3]. In the example the FC switches and devices from Austin, New York, and Los Angeles form a single extended high latency fabric. Switch to switch communication traffic must be converted and sent across the internet just like device to device storage traffic.

iFCP is an individual device proxy protocol which maintains individual SAN islands as separate entities. In the example each city would have an individual FC SAN. The switch to switch traffic in the Austin SAN would not be sent to New York or Los Angles. The iFCP gateway presents itself as an edge switch on the local SAN. Remote devices appear to the local fabric as though they are attached to the gateway.

A DSAN is typically used for replication or backup purposes which require sustained bulk data transfer between end devices. The parameter selected to measure the performance of DSAN is the throughput across the internet link. Since FC-to-IP involves encapsulating FCP data into an IP packet, the TCP implementation play a significant role on the performance of a DSAN. The TCP implementation is housed in the gateways in Figure 1. The links between the gateways are typically operated at OC3 speeds although only a significant minority of deployments actually utilizes the full Gigabit-Ethernet speed. Even though most deployed links do not experience sustained high packet loss rates, the storage traffic has to maintain throughput when packet losses do occur or during transient problems. The maximum distance and possible throughput depends primarily upon the internet packet loss and latency; gateway's buffering and TCP/IP stack implementation. The effect of latency and packet loss on performance is dependent on the TCP congestion control techniques, window scale options, error detection and recovery techniques. The internet packet loss on an average across continental US is 16% with an average delay of 67 msecs [http://www.internettrafficreport.com] indicating multiple packet loss. The internet link between the FCtoIP gateways may also consist of wireless sections leading to higher packet loss rate (in excess of 25%). Since, DSAN implementation is achieved by leasing inter continental high speed links (OC3) from service providers, in this investigation, the effect of wireless links is not considered. The authors recognize the existence of vast literature discussing the packet loss issue in wireless networks. At present, majority of bulk transfer operations are performed at enterprise data centers implemented on FC SAN constituting mainframes and high end servers. Even though iSCSI is becoming popular in small data centers due to low infrastructure costs and low incompatibility issues, it has not been widely implemented at enterprise data centers. Hence, the modifications to TCP/IP stack in iFCP, to improve the throughput during bulk transfer with large latency and packet loss is the focus of this investigation. Also, it should be noted that the modifications to the TCP/IP stack investigated can be applied to FCIP and iSCSI with minimal additional work.

This paper first presents a theoretical study of the effects of some of the modifications of the TCP/IP stack implementation known as storage friendly TCP/IP on high rate, long latency sustained bulk storage traffic performance. Next, the performance due to the modifications is demonstrated by experimental results obtained using iFCP gateway's with the modified TCP/IP stack.

2. RELATED WORK

A large body of literature exists regarding modification to TCP/IP stack to improve the performance under packet loss and congestion scenarios. But, the literature is very sparse with regard to modifications to TCP/IP stack in high latency links in particular with distributed storage area networks. Hence a few investigations related to this investigation are discussed. The investigations [4, 5, 6, 7, 8 and 9] discuss improving the performance by modifying the TCP/IP stack either under congestion conditions or large latency. The investigation [4] discusses a delay based approach to eliminate the congestion control problem associated with the binary nature of congestion control. The investigation [5] discusses enhancing the IP routing capability by combining the congestion control with routing scheme. In the third investigation [6], the modifications to improve performance when packet reordering occurs due to route fluttering and retransmissions are discussed. A number of algorithms have been proposed in this investigation. The fourth investigation [7] deals with hybrid networks consisting of terrestrial and wireless links. This investigation deals with improving the performance over the wireless links by TCP splitting. The common thread of all of the above investigations is that the links have low to medium latency and the desired data transfer rates are in the range 10 to 100 Mbps. As mentioned previously, DSANs experience large latency (>100 ms) at data transfer rates of 1 to 4 Gbps. Also, the investigations in cited literature have demonstrated the improvement by theoretical analysis in most cases with simulations in a few [8, and 9]. In this investigation, we not only show the improvement through theoretical analysis but also through experimental results by implementing the modified TCP/IP stack on iFCP gateways.

3. iFCP

A DSAN based on iFCP is shown in Figure 2. In this figure, local FC devices attach directly to F_Ports on the gateways. Only device-to-device storage traffic is allowed to pass through the gateways creating SAN islands. For example if one of the gateway's ports in the diagram were instead an E_Port attached to a local switch, such a switch would not be allowed to communicate through the gateway, but storage traffic passing through the switch destined for a remote device would be forwarded across the IP network. This behavior ensures that any disruptive behavior within a local SAN is contained improving DSAN stability. The remote devices are addressed in two modes in iFCP: transparent or translated. Transparent addressing in iFCP is rarely used in practice and so will not be addressed in this paper.

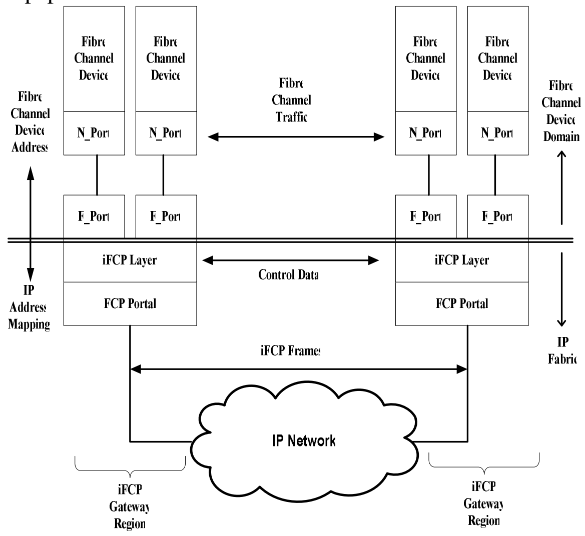

Figure 2. iFCP based DSAN [11, 12]

The iFCP address translation is analogous to IP network address translation, where an iFCP gateway presents remote devices as though they are directly attached to the gateway. The gateway appears as an FC switch to the local fabric. The iFCP gateway assigns local 24-bit FC IDs known as the proxy address to the remote device N_Ports [1, 12]. In addition iFCP always assigns an iFCP specific FC ID to the device and uses this ID to distinguish device and its traffic within a group of gateways. iFCP gateways perform address conversion when they convert from FC protocol to iFCP protocol.

An iFCP gateway communicates across an IP network using an assigned IP address and TCP/IP connections. The gateway opens a separate TCP/IP connection for each local FC device to remote FC device session created. Each of these sessions uses the gateway IP address and the iFCP FC IDs assigned by the gateways to keep track of the TCP connections and storage traffic. This enables communication between any pair of FC devices over the IP network. Since each device pair communication session has a TCP connection, an iFCP frame is routed to its destination using standard IP infrastructure and the gateway performs congestion control, error detection and recovery through TCP. The iFCP gateways maintain a lookup table [12] containing actual remote N_Port addresses, proxy N_Port addresses and IP addresses. On receipt of an FC frame destined for a remote device, the iFCP gateway translates the local source device's FC address into an iFCP address, translates the remote device's proxy FC address into an iFCP address, encapsulates the FC frame in an iFCP header, determines which TCP/IP connection belongs to that FC session, and transmits the encapsulated frame on the TCP/IP connection. The IP header in the iFCP frame will have its destination address set to the destination iFCP gateway IP address. When the iFCP gateway receives a TCP segment, the gateway determines if it can complete the construction of an encapsulated FC frame for that connection. If it can, the iFCP gateway translates the destination iFCP address into the destination device's local FC address and the source iFCP address into the remote devices proxy address. It then forwards the FC frame into the local SAN.

Translation of addresses improves scaling and configuration of DSANs. Since remote devices are imported as individual devices independent of their actual local FC SAN attachment or addressing, multiple remote devices can have the same FC addresses on their individual remote SANs and yet still be uniquely accessible from a the local SAN through their proxy addressed.

4. STORAGE FRIENDLY TCP/IP

The storage friendly TCP/IP is a group comprising a number of modifications to the TCP/IP stack designed to increase data transfer rate with packet loss and large latency. The major modifications implemented were TCP Window Scaling, fast retransmit/recovery, reorder resistance, and selective acknowledgement scheme. Due to page limitations, the modifications of storage friendly TCP/IP that will be discussed in this paper are the automatic TCP window scaling, and the modified fast retransmit/recovery algorithm.

Slow Start and TCP Window Scaling

The TCP connection carrying iFCP traffic typically runs across a long fat network (LFN) [13, 14] with a large capacity. The capacity of LFNs is expressed mathematically as the product of bandwidth (BW) and round-trip time (RTT) as shown in (1).

$$\text{Capacity (bytes)} = \text{BW (bytes/sec)} \times \text{RTT (sec)} \qquad (1)$$

This capacity, often called the bandwidth delay product, represents how much data the link or pipe can hold at a given time. To achieve maximum throughput for a given latency, the sender must transmit data at the capacity of the pipe. When this is achieved it is known as the ideal steady state of the connection. For TCP/IP the maximum amount of data that can be transmitted by a sender without waiting for an acknowledgement from the receiver is dependent on the advertised TCP receive window (TCP receive buffer). The initial value for this window and its scale value are determined during connection establishment [15].

As shown in Figure 2, an iFCP TCP connection can cross an arbitrary IP network. This network can include multiple sub-networks with potentially slow links or congestion. If the sender starts off by immediately injecting multiple TCP segments (packets) up to the TCP window size advertised by the receiver this can lead to drastic reduction of throughput [13] due to packet loss in slow routers and links along the network path followed by the iFCP connection. In order to reach the ideal steady state without drastic reduction in throughput, TCP uses an algorithm known as slow start [13, 15]. The slow start algorithm states that the rate at which new packets should be injected into the network is the same as the rate at which the packets are acknowledged. This leads to exponential growth of the amount of data that the sender is allowed to transmit. The sender's TCP implementation tracks this amount per connection with another window known as congestion window (cwnd). During the connection setup process, the cwnd is initialized to one segment. The slow start algorithm increases this cwnd by one segment each time an acknowledgement (ACK) is received. The sender can now transmit up to the minimum of the current cwnd and the advertised TCP receive window.

In Figure 3, the interactions between the sender and receiver at discrete time steps to transfer certain amount of data using the slow start algorithm is shown. For illustration purposes, assume that one time unit is needed to transmit one segment and also assume that the receivers advertised window is large enough that it is not a limit for the example.

At time 0, the sender's cwnd is 1 segment and the sender transmits a single segment D1. D1 travels to the receiver at time steps 1 and 2. At time 3 the receiver gets D1 and acknowledges it with transmission A1 at time 4. A1 travels back to the sender at times 5 and 6 (not shown). At time 7, the sender gets A1, increases the cwnd to 2 segments, and notes that the RTT is 8 time units. At time 8, the sender transmits D2 and then at time 9 transmit D3 since the sender is allowed to transmit 2 segments by the current cwnd. These then travel to and are acknowledged by the receiver such that by time 16 the sender has a cwnd of four and can transmit D4, D5, D6, and D7 into the network. At times 23, 24, 25, and 26 the sender increases the cwnd by 1 and transmits another segment with a cwnd of 8 with 4 segments already transmitted (D8, D9, D10, and D11) at time 27. The sender can then send 4 additional segments D12, D13, D14, and D15 without violating the cwnd value before A8 is received at time 31. It can be seen that by time 31, the number of segments entering the pipe is equal to the number of ACKs returning indicating the ideal steady state of the connection. At this time, the cwnd equals or exceeds the advertised TCP receive window or capacity of the network, causing the data flow to be limited. The term ideal here represents a condition where there are no packet losses.

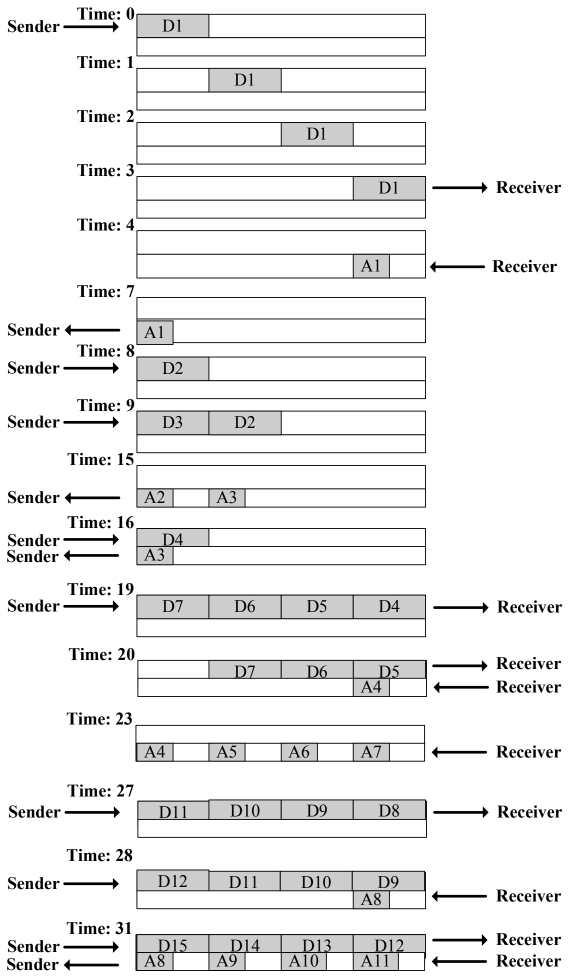

Figure 3. Slow start with bulk data transfer [13]

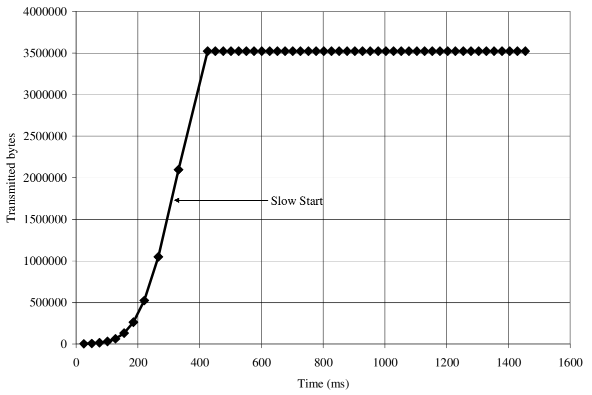

Figure 4. Continuous data flow with slow start

The continuous time data flow due to the slow start algorithm and the advertised TCP receive is shown in Figure 4. The exponential increase represents the slow start, and the constant part represents the pipe reaching the ideal steady state. The ideal steady state value depends on the capacity of the pipe and the advertised TCP receive window size. If the advertised TCP receive window size is less than the capacity of the pipe, the throughput achieved will be less than the bandwidth of the pipe. The standard TCP implementation [16] allows a receiver to advertise a maximum of 64 KB TCP window due to the use of only 16-bits in the TCP header for this purpose.

Typically, iFCP runs through networks formed by interconnection of local area networks (LANs) using high speed backBone networks operating at data rates in excess of 1 Gbps (125 MB/s) and possibly experiencing round-trip time in excess of 100 msecs[10]. To achieve maximum throughput on this kind of LFN, the advertised TCP window has to be significantly higher than the maximum 64 KB possible in the standard TCP. In this work, the TCP implementation at the receiver has been modified to implement TCP window scaling option [17, 18]. The TCP window scaling option involves increasing the size of the window covered with the 16-bit TCP header window value by applying a 1 byte scale factor to the value. The scale factor indicates by how many bit positions the 16-bit TCP window value in the TCP header has to be left shifted to obtain the real advertised window size. The scale factor is sent as an option parameter in the TCP header during connection establishment. Using the scale factor [13] the theoretical maximum advertised window size has been increased from 64KB to 64KB x 2^{14}. The received scaling factor is stored at the iFCP gateways. The iFCP gateway implementation used for the tests in this paper always negotiates a scale factor of 10 for its iFCP TCP connections giving 1 KB per count in TCP header window size field.

Modified Fast Retransmit/Recovery Algorithm

A congestion avoidance algorithm to handle packet losses is an integral part of standard TCP implementation [16]. The implementation details and analysis of the congestion avoidance algorithm has been discussed extensively in various literatures [13, 14, and 19]. If a packet loss is detected due to the expiration of the retransmission timer, half of the current cwnd is stored in the slow start threshold (ssthresh), the cwnd is reinitialized to one segment, and transmission resumes (using slow start) starting with any unacknowledged data. The transmission of data continues under slow start only up to a cwnd of ssthresh (which is half of the original cwnd). Above this value cwnd only increases by one segment per RTT regardless of how many ACKs are received. This changes the exponential growth of cwnd into linear growth and is known as congestion avoidance algorithm [13]. This process is repeated any time the transmission timer expires (on any packet loss) and can cause significant reduction of throughput. Later, the TCP implementation was modified to retransmit the missing segment before the retransmission expired, cut the current cwnd in half, and then enter congestion avoidance [14, 15, and 18] but without entering slow start. Entering slow start would force the cwnd to 1 segment and force the retransmission of all unacknowledged data. This retransmission of the missing segment followed by congestion avoidance is known as fast retransmit and fast recovery algorithms. The fast retransmit algorithm is based on the requirement that TCP has to generate an immediate ACK (a duplicate ACK) when an out-of-order segment is received. This duplicate ACK is used to inform the sender that a segment was received out-of-order, and to tell what sequence number is expected. Since, the duplicate ACK could be caused either by packet loss or simple reordering; the sender waits for a small number of duplicate ACKs to be received. In case of only reordering of the segments, this will cause only one or two duplicate ACKs before the reordered segment is processed, which results in a new (non-duplicate) ACK [13]. Three or more duplicate ACKs received by a sender in a row, indicate strongly the loss of a segment and initiates the fast retransmit algorithm without waiting for the transmission time to expire. The fast retransmit and fast recovery algorithm stages implemented together are shown below:

1. On receiving the third duplicate in a row, set ssthresh to one-half of the minimum of current cwnd and the advertised TCP receive window.
2. Retransmit the missing segment.
3. Set current cwnd to ssthresh plus 3 times the segment size.
4. Each time another duplicate ACK arrives, increment cwnd by the segment size and transmit a packet (if the new cwnd is less than the advertised TCP receive window).
5. Next, on receiving an ACK acknowledging new data, set current cwnd to ssthresh (one half of the cwnd when packet loss was detected).

Figures 5 and 6 show the simulated data flow under packet loss with slow start/congestion avoidance and fast retransmit/fast recovery algorithms.

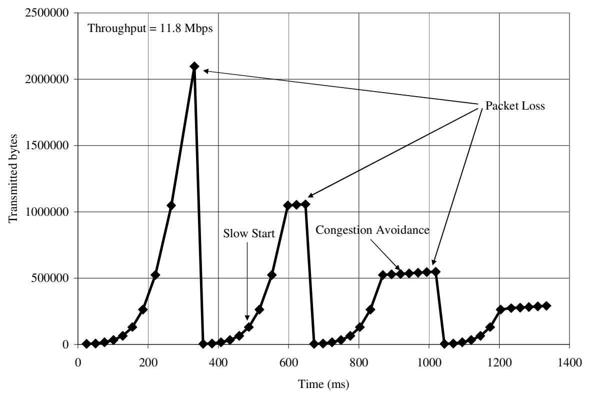

Figure 5. Data flow under packet loss with slow start/congestion avoidance

In Figure 5, because fast retransmit and fast recovery are not used, the cwnd is initialized to one segment every time a packet is lost and slow start is initiated before congestion avoidance.

In Figure 6, fast retransmit prevents slow start from occurring when a packet is lost. The effects of fast recovery and congestion avoidance are also shown. The overall throughput is higher in this case as compared to figure 5. The fast transmit and fast recovery algorithm allows higher throughput as compared to only slow start under moderate congestion. However, the iFCP connections in a DSAN with very large TCP receive window capability (several MB) at the iFCP gateways are expected to provide close to maximum network throughput even with packet losses. To meet this requirement, the fast retransmit/fast recovery algorithm has been modified. The standard fast retransmit/fast recovery algorithm reduction of the current cwnd value at the time of packet loss detection to one-half of its value as seen in Figure 6 contributes to reduced throughput. Based on this observation, the fast retransmit/fast recovery algorithm was modified to reduce the current cwnd value at the time of packet loss detection by only 1/8th of its current value. This causes the TCP implementation to react more slowly to packet loss events. Figure 7 shows the simulated data flow under packet loss with the modified fast recovery algorithm. From the graph, it can be seen that the sender is responding slowly to packet loss there by maintaining a higher throughput.

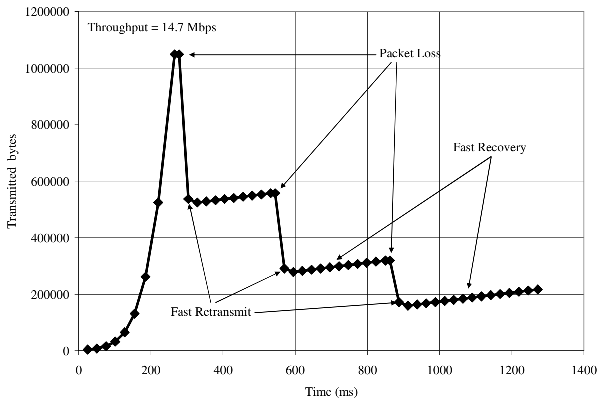

Figure 6. Data flow under packet loss with fast retransmit/fast recovery

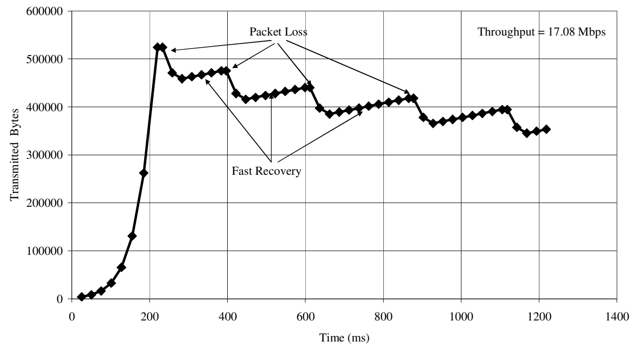

Figure 7. Data flow under packet loss with smaller cwnd reduction fast retransmit/fast recovery

As can be seen from the figures the linear ramp of cwnd after a packet loss event suppresses throughput. An option to disable congestion avoidance was also added so that the ramp of cwnd would always be exponential.

5. EXPERIMENTAL SETUP

The experimental setup used to measure a DSAN performance is shown in Figure 8. FC traffic is generated by using FC simulator (FCSim) cards instead of actual FC end devices (server and storage). An FCSim card manufactured by Brocade can generate FC traffic at rates up to 4 Gbps using the embedded FCLoad^{TM} program. In this test setup, two FCSim cards are located in a single PC, with one FCSim card generating traffic and the other receiving the traffic. An FCSim card can also measure throughput and the latency of every FC frame transmitted and received and was used to collect the performance data. The FCSim cards are connected to two Sphereon 4500 FC switches from Brocade. The two FC switches in turn are connected to two Eclipse 1620 iFCP gateways from Brocade Corp. The two iFCP gateways are connected together by a 1 Gbps IP link through a packet loss/latency generator. The packet loss/latency generator is used to simulate packet loss and RTT conditions in the IP network. The modified TCP/IP stack is implemented on the two iFCP gateways by programming the firmware on the two Eclipse 1620s. The Eclipse 1620 TCP implementation has a maximum TCP receive window size per connection of 8MB using a scale factor of 10.

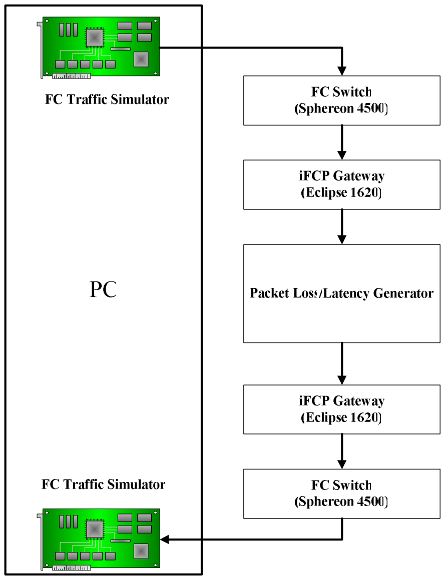
FC Traffic Simulator
FC Switch (Sphereon 4500)
iFCP Gateway (Eclipse 1620)
PC
Packet Loss/Latency Generator
iFCP Gateway (Eclipse 1620)
FC Traffic Simulator
FC Switch (Sphereon 4500)

Figure 8. DSAN test setup

6. EXPERIMENTAL RESULTS

The theoretical throughput possible with increasing RTT at a constant 8MB TCP window size on a 1 Gbps iFCP link is presented in Table 1, this information was computed using (1). Using the previously discussed experimental setup, the FCLoad program was used to transmit traffic at data rates (load) ranging from 10% to 90% of the link bandwidth. The packet loss/latency generator was used to simulate RTT ranging from 0 to 200 ms to simulate internet behavior [10].

Table 1 Theoretical throughput with increasing RTT

TCP Window Size (MB)	Round Trip Delay (ms)	Computed Throughput (MB/s)
8	80	100.00
8	100	80.00
8	120	66.67
8	140	57.14
8	160	50.00
8	180	44.44
8	200	40.00

Figure 9 shows the variation of throughput with increasing RTT. It can be seen that up to 80 ms RTT, increasing the load causes a corresponding linear increase in the throughput. When the RTT is increased to 100 ms, it can be seen that the throughput can reach only 80 MB/s due to the advertised TCP window size maximum value of 8MB. As the RTT is increased further, the limiting effect of the TCP window size further reduces the throughput. The measured steady state throughput shown in Figure 9 matches the theoretical values shown in Table 1.

Next, FCLoad was used to drive 90% of line rate with the network simulation applying a constant RTT of 25 ms and various loss rates. In this setup the effects of the fast recovery modification and congestion avoidance disabling were measured. Figure 10 shows these results. It can be seen that the throughput with standard TCP decreases to a value of 9 MB/s at 1 in 1000 packet loss rate. The throughput with the same packet loss rate with smaller cwnd reduction modification and congestion avoidance disable has increased to 16MB/s and 11.5MB/s respectively. The increase in throughput with congestion avoidance disable modification by itself is less compared to the smaller cwnd reduction.

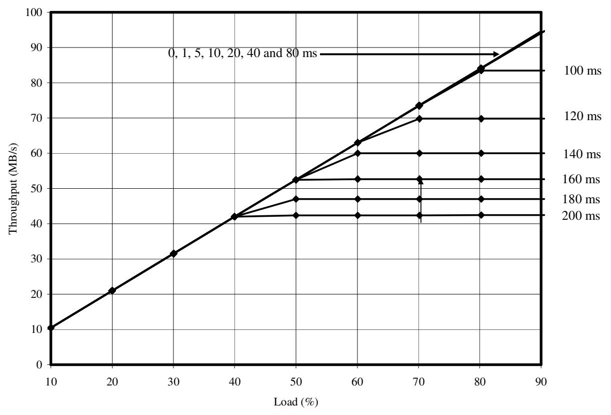

Figure 9. Throughput variation with increasing RTT and constant 8MB TCP window size

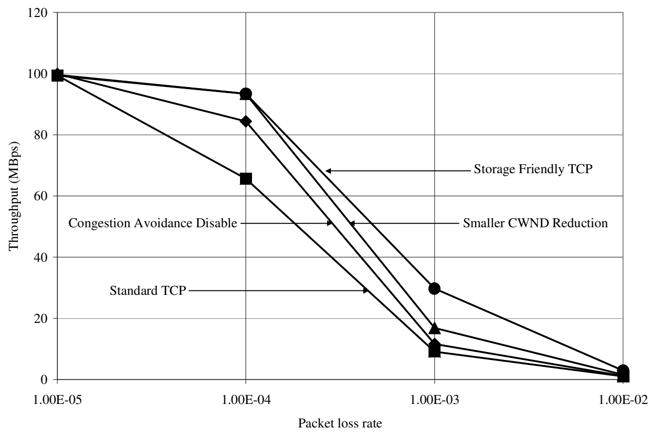

Figure 10. Storage friendly TCP influence on throughput with packet loss rate

This is due to the congestion avoidance disable starting point being at one-half of the cwnd value as compared to 7/8th of the cwnd value with smaller cwnd reduction. Next, on selecting both of the modifications the throughput with the same packet loss rate has increased to a value of 30MB/s. This is due to exponential ramp from the higher 7/8th cwnd value.

7. CONCLUSIONS

The problems faced with bulk storage data transfer with the standard TCP implementation has been demonstrated both by theoretical analysis and experimental results. The standard TCP implementation performance has been improved with LFNs by incorporating the TCP window scaling option. The improvement in throughput by using fast retransmit/fast recovery over slow start/congestion avoidance has been demonstrated with a theoretical analysis. Further modifications of fast retransmit/fast recovery by decreasing the one-half cwnd reduction to 1/8th cwnd reduction on detecting a packet loss increase the throughput significantly. This modification when paired with the congestion avoidance disable increases the throughput by a factor of 3.

The modifications introduced to the standard TCP and know as storage friendly TCP or storage optimized TCP enable more widespread implementations of iFCP based DSANs across transcontinental distances for storage backup and replication.

Further modifications such as reorder resistance, increasing initial cwnd value in slow start, and others remain to be explored.

8. REFERENCES

[1] R. W. Kembel, *Fibre Channel Switched Fabric*, Northwest Learning Associates, Inc., 1st Edition, 2001.

[2] S. Muknahallipatna, T. Brothers, N. Mandagere, P. Patil, and J. Hamann, "The effect of end to end latency in a distributed storage area network on Microsoft Exchange Server 2003 performance - part 1," *The 29th Annual IEEE Conference on Local Computer Networks (LCN)*, 16-18 November 2004.

[3] *Fibre Channel over TCP/IP (FCIP)*, RFC 3821, The Internet Engineering Task Force (IETF)**,** July. 2004.

[4] D. Wei, C. Jin, S. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Transactions on Networking, Vol. 14, No. 6*, December 2006, 1246-1259

[5] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet," *IEEE/ACM Transactions on Networking, Vol. 14, No. 6,* December 2006, 1260-1271

[6] D. Leung, V. LI, and D. Yang, "An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges," *IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 4*, April 2007, 522-535

[7] J. Zhu, S. Roy, and J. Kim, "Performance Modelling of TCP Enhancements in Terrestrial-Satellite Hybrid Networks," *IEEE/ACM Transactions on Networking, Vol. 14, No. 4*, August 2006, 753-766

[8] J. Jo, Y. Kim, and H. Chao, "TCP Performance Comparison Under Various Load Balancing Methods Using OPNET," *in Proc. OPNETWORKS*, August 2002

[9] G. Corral, A. Zaballos, T. Fulgueira, and J. Abella, "Simulation-based study of TCP flow control mechanisms using OPNET Modeler," *in Proc. of OPNETWORKS*, August 2002

[10] AT & T Global IP Network, Network delay, http://ipnetwork.bgtmo.ip.att.net/pws/network_delay.html

[11] T. Clark, IP SANs: *A Guide to iSCSI, iFCP and FCIP Protocols for Storage Area Networks,* Addison-Wesley, 1st Edition, 2002.

[12] *A Protocol for Internet Fibre Channel Storage Networking - iFCP*, RFC 4172, The Internet Engineering Task Force (IETF)**,** September 2005.

[13] W. R. Stevens, *TCP/IP Illustrated: The protocols*, Vol. 1, Addison Wesley, 1994.

[14] *TCP Extensions for High Performance*, RFC 1323, The Internet Engineering Task Force (IETF), May 1992.

[15] *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, RFC 2001, The Internet Engineering Task Force (IETF), January 1997.

[16] V. Jacobson, "Congestion Avoidance Algorithm," *in Proc. ACM SIGCOMM*, 1988, pp. 314-329.

[17] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Explicit Window Adaptation: A Method to Enhance TCP Performance," *IEEE/ACM Trans. on Networking, Vol. 10, No. 3, June 2002*.

[18] T. V. Lakshman, and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. on Networking, Vol. 5, No. 3, June 1997*.

[19] G. R. Wright, and W. R. Stevens, *TCP/IP Illustrated: The Implementation*, *Vol. 2, Addison Wesley, 1994*.