Evaluation of Advanced TCP Stacks in the iSCSI Environment using Simulation Model

Girish Motwani
Indian Institute of Science, Bangalore
girish@csa.iisc.ernet.in

K. Gopinath
Indian Institute of Science, Bangalore
gopi@csa.iisc.ernet.in

Abstract

Enterprise storage demands have overwhelmed traditional storage mechanisms and have led to the development of Storage Area Networks (SANs). This has resulted in the design of SCSI transport protocols that encapsulate SCSI commands and data for transfer over the network. Fiber channel protocol was the first such protocol that used Gigabit per second speed links to carry SCSI commands and data over long distances. However, with the emergence of Gigabit Ethernet and the iSCSI (Internet SCSI) protocol that maps the SCSI block oriented storage data over TCP/IP and enables storage devices to be accessed over standard Ethernet based TCP/IP networks, reduction in costs and a unified network infrastructure can be achieved.

The iSCSI data flow is regulated by the TCP congestion control algorithm. The standard TCP Reno congestion control algorithm substantially under utilizes the network bandwidth over high speed connections for most applications. To address this limitation of TCP, variants of the TCP congestion control algorithm, designed for high bandwidth networks, such as FAST TCP, BIC-TCP, H-TCP, and Scalable TCP have been proposed. We use simulations and experiments to compare the performance of these TCP variants in an iSCSI environment. Our results indicate that H-TCP outperforms the other TCP variants in an iSCSI environment. However H-TCP results in unfair sharing of network bandwidth when simultaneous multiple flows exist. Performance obtained using BIC-TCP is only second to that using H-TCP and it is relatively fairer as compared to H-TCP.

1. Introduction

Enterprise storage demands have overwhelmed traditional storage mechanisms and has led to the development of SANs. In a SAN, the storage devices are directly attached to the network thereby enabling multiple host computer systems to access the storage devices using standard network protocols. With the advent of Gigabit per second and higher speed networks, the effective transfer rates between the host and the storage have become comparable to those achieved with direct attached storage. This has accelerated the efforts directed towards the design of SCSI transport protocols that encapsulate SCSI commands and data for transfer over the network.

Fiber channel protocol was the first such protocol that used Gigabit per second links to carry SCSI traffic over distance up-to 10 Kms. Although Fiber channel implementations offer better throughput guarantees, they are subject to distance limitations and compatibility problems. To alleviate these problems, the iSCSI protocol that maps the SCSI block oriented storage data over TCP/IP and enables storage devices to be accessed over standard Ethernet based TCP/IP networks has been proposed. iSCSI establishes a communication session between the SCSI Endpoints. This session may comprise of one or more TCP connections.

The iSCSI protocol would typically be used in large bandwidth network environments. It has been reported in [8], that the standard TCP Reno congestion control algorithm substantially under utilizes the network bandwidth over high speed connections. In the congestion avoidance phase, the TCP Window increases by 1 every round trip time(RTT) and reduces by half at a loss event. This places a serious constraint on the congestion window that can be achieved by TCP in realistic environments. For example, following a loss event, to achieve full utilization of 1Gbps with 1500 bytes packets, the standard TCP variant (Reno) requires approximately 8300 RTTs. With a 100ms RTT, it takes approximately 14 minutes. Further, the average packet drop rate of at most $2 * 10^{-8}$ (obtained using the relation $windowsize = 1.2/\sqrt{p}$, where p is the loss probability) needed for full link utilization in this environment corresponds to a bit error rate of at most $2 * 10^{-12}$,which is an unrealistic requirement for high speed networks. To address this fundamental limitation of TCP , variants of the TCP congestion control algorithms such as FAST TCP, H-TCP, BIC-TCP and Scalable TCP have been proposed. We implemented the iSCSI simulation model using network

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

simulator ns-2[4] for simulating the network behavior and DiskSim[7] for simulating the disk behavior to compare the performance of these TCP variants in an iSCSI environment. To validate the simulation model, we conducted real performance measurements and compared the results thus obtained with those obtained using simulations. In addition, we study the effect of some iSCSI parameters on the performance obtained.

The rest of the paper is organized as follows. In Section 2, we provide details on the iSCSI protocol data transfer. The various advanced TCP congestion control algorithms that we compare in our study are described in Section 3. Section 4, describes our implementation of the iSCSI protocol in ns-2. The Experimental setup is described in Section 5. We present our results in Section 6. Related work is discussed in Section 7 and Conclusions appear in Section 8.

2. iSCSI Protocol Data Transfer

The iSCSI protocol is a SCSI transport protocol that maps the SCSI block oriented storage data over TCP/IP and enables storage devices to be accessed over standard Ethernet based TCP/IP networks. The iSCSI protocol stack is as shown in figure 3. iSCSI establishes a communication session between the initiator and target. The session may consist of one or more TCP connections. SCSI Command Descriptor Blocks (CDB) are passed from the SCSI layer to the iSCSI transport layer. The iSCSI transport layer encapsulates the SCSI CDB in an iSCSI Protocol Data Unit (PDU) and forwards it to TCP. On a receive, the iSCSI transport layer extracts the CDB from the iSCSI PDU, received from the TCP layer and forwards the CDB to the SCSI layer.

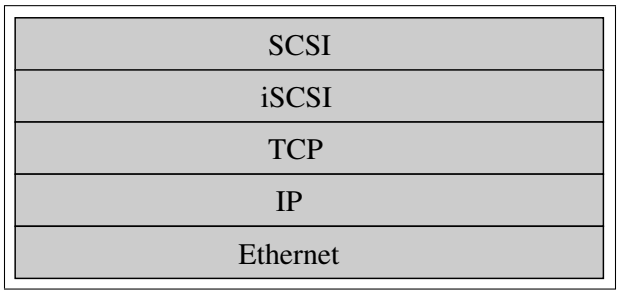

Figure 1. The iSCSI Protocol Stack

To illustrate the working of the protocol, consider a write request. A SCSI write request triggers the transmission of iSCSI command PDU to the target. The target on receiving the command, allocates buffers for the transfer and responds with one or more (R2T) PDUs. Each R2T PDU is a permission to the initiator to transfer a portion of the data associated with the command. The initiator responds to a R2T PDU by sending out a sequence of Data-Out PDUs containing the data requested. The maximum amount of data that can be sent out in one PDU is given by the target's Maximum Receive Data Segment Length(MaxRecvDataSegmentLength). Finally when all the data for the command, has been transferred from the initiator to the target, the target sends an iSCSI Response PDU, indicating successful completion of the command. The iSCSI layer at the initiator passes the command completion status information to the SCSI layer. The write transfer is illustrated in figure 2

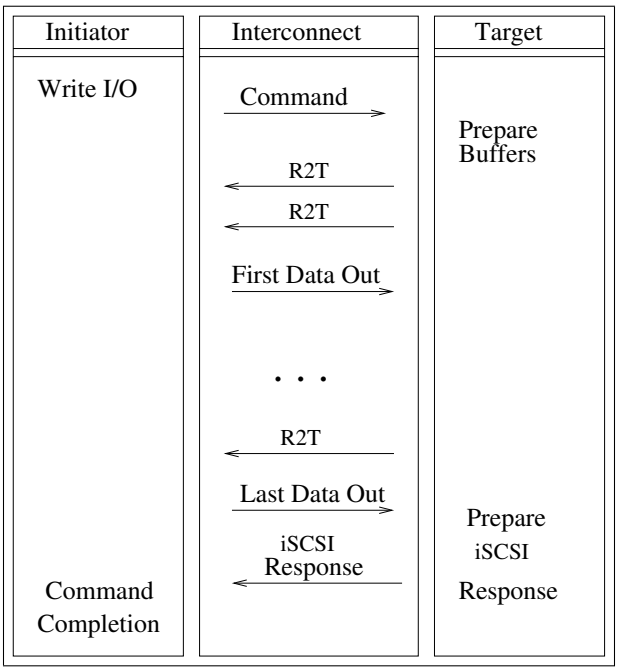

Figure 2. iSCSI Write Transfer

2.1. iSCSI and TCP

Storage traffic requires reliable and in-order delivery of data packets. Since TCP guarantees in-order delivery of data and congestion control, the iSCSI protocol was designed to work on top of TCP as shown in Figure 1. TCP has a mechanism to acknowledge all TCP packets that are received and to resend packets that are not acknowledged within a given time-out period. Thus iSCSI packets sent over TCP that may get lost during delivery are automatically resent by TCP. The iSCSI data flow is regulated by the congestion control mechanism of TCP.

As TCP is used as the transport for iSCSI, the iSCSI initiators are connected to targets using TCP connections. Due to the TCP window size restrictions and round trip times over long distances, it might not be possible for a

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

single TCP connection to utilize the full bandwidth capacity of the underlying link. Therefore, the iSCSI protocol specifies that an iSCSI session can consist of multiple TCP connections between the initiator and target.

3. TCP Variants

TCP has been adopted as a data transfer protocol for the Internet. However, it has been reported that over high speed networks, TCP under utilizes the network bandwidth. A number of variants of the TCP congestion control algorithm have been proposed to circumvent this problem. In this section, we describe the standard TCP Reno algorithm and the approach taken by the proposed TCP variants.

3.1. Reno TCP

TCP's congestion management comprises of the slow start and congestion avoidance algorithms that allow TCP to increase the data transmission rate without overwhelming the network. TCP Reno's congestion avoidance mechanism is referred to as AIMD(Additive Increase, Multiplicative Decrease). In the congestion avoidance phase, TCP Reno increases its congestion window(cwnd) by one packet per window of data acknowledged and halves the congestion window for every window of data containing a packet drop. This can be described by the following equations.

Slow Start:

ACK : $new\_cwnd = old\_cwnd + c$

Congestion Avoidance:

ACK : $new\_cwnd = old\_cwnd + a/old\_cwnd$

Loss : $new\_cwnd = old\_cwnd - b * old\_cwnd$

where a=1, b=0.5, c=1.

3.2. FAST TCP

FAST TCP[8] is a TCP congestion control algorithm designed for high speed, long latency networks. It is based on TCP Vegas instead of Reno TCP. The key difference is that it uses an equation based window control approach rather than the AIMD algorithm of TCP Reno. Also, unlike TCP Reno, it uses both queuing delay and packet loss as the congestion measures rather than just packet loss.

3.3. BIC-TCP

BIC-TCP[14] is another variant of the TCP congestion control algorithm designed for high speed networks with large delays. Like TCP Reno, BIC-TCP uses packet loss as the congestion measure. However, it uses the binary search technique to increase the congestion window in the congestion avoidance phase.

3.4. Scalable TCP (STCP)

Scalable TCP involves a simple sender side change to TCP Reno. [10] defines the legacy window size *lwnd* as the maximum window size that can be achieved by TCP Reno. Associated with this window size is the legacy loss rate p_l which is the maximum packet loss rate needed to support window larger than *lwnd*. Scalable TCP uses the Reno congestion window update algorithm given as

ACK : $new_{cwnd} = old_{cwnd} + 1/old_{cwnd}$

LOSS : $new_{cwnd} = old_{cwnd} - [0.5 * old_{cwnd}]$

when $cwnd_{old} \leq lwnd$.

When $cwnd_{old} > lwnd$ the following Scalable TCP window update algorithm is used.

ACK : $new_{cwnd} = old_{cwnd} + 0.01$

LOSS : $new_{cwnd} = old_{cwnd} - [0.125 * old_{cwnd}]$

3.5. High Speed–TCP (HSTCP)

High Speed TCP (HS-TCP) [6] is designed to behave like Reno for small values of the congestion window, but above a chosen value of *cwnd* an aggressive response function is used. When *cwnd* is large (greater than 38 packets), this modification uses a table to determine by how much the congestion window should be increased when an ACK is received, and it releases less network bandwidth than $\frac{cwnd}{2}$ on packet loss.

3.6. H-TCP

H-TCP has a similar approach to HSTCP since H-TCP switches to the advanced mode after it has reached a threshold. Instead of using a table like HS-TCP, H-TCP uses a heterogeneous AIMD algorithm as described in [13]

4. iSCSI Simulation model in ns-2

We have implemented the iSCSI protocol in network simulator ns-2 which provides substantial support for simulation of routing, multi cast protocols and transport protocols, such as UDP, TCP, over wired and wireless networks. To simulate the disk behavior, we have integrated the DiskSim Simulator with ns-2 for our simulation setup. DiskSim acts as the slave of the system simulator(ns-2).

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

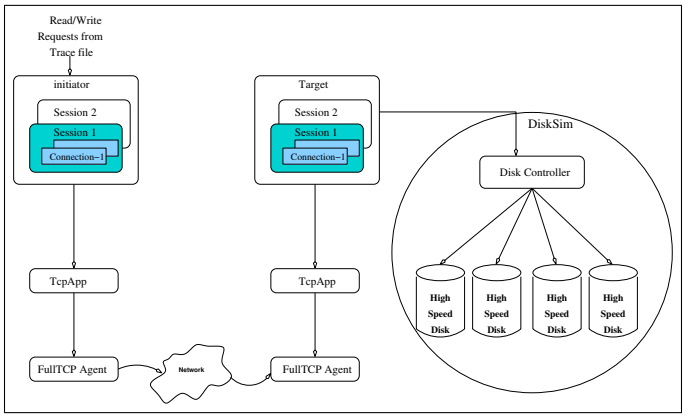

Figure 3. iSCSI System based on ns-2

4.1. Implementation

Our implementation of the iSCSI protocol in ns-2 supports the following features

- Single session between the initiator and target
- Multiple connections per session
- Arbitrary number of outstanding R2Ts
- All combinations of InitialR2T and ImmediateData keys
- arbitrary values of MaxBurstLength, FirstBurstLength and MaxRecvDataSegmentLength
- optional periodic Nop-Ping to target

The iSCSI protocol implementation in ns-2 comprises the following classes:

iSCSIData: This class is used to encapsulate the iSCSI Protocol Data Units to be sent across to the other side using the TCPApp class object associated with the connection.

iSCSIApp: This is the base class for the iSCSI Initiator and Target. The iSCSI parameters are: nconnections, MaxConnections, InitialR2T, ImmediateData, MaxBurstLength, FirstBurstLength, MaxOutstandingR2T, DataPDUInOrder.

iSCSIConnection: This class performs tasks specific to the iSCSIConnection. Each iSCSIConnection has its own TcpApp object which is used for the transmission and reception of data. FullTCPAgents are used for iSCSIConnection.

iSCSISession: This class performs functions specific to the iSCSISession operation. The iSCSIConnection class object is an element of this class.

iSCSIInitiator: This class is derived from the iSCSIApp class and includes the functions specific to the initiator operation. This includes the iSCSISession Class object as an element.

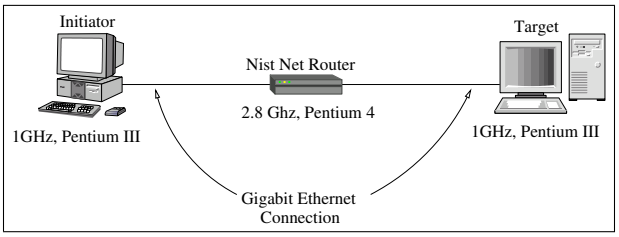

Figure 4. Experimental Setup

iSCSITarget: This class is derived from the iSCSIApp class and includes the functions specific to the iSCSI Target operation. It also includes the iSCSISession Class object as an element.

Figure 3 shows a typical setting of the iSCSI system implementation in ns-2. The system comprises of four parts: the initiator, which generates the read/write requests, the network, configured based on the ns-2 components, the iSCSI target and the disk devices simulated using DiskSim. The initiator is fed with traces provided by Hewlett-Packard Labs.

5. Experiments

We conducted our experiments using both the Linux implementation of the iSCSI initiator and target[11] and simulation setup using ns-2 and DiskSim. In this section, we describe the setup used.

5.1. Testbed

The WAN emulation testbed is as shown in Figure 4. Three machines were used in our experimental setup. The machines that host the initiator and target have 1 GHz Pentium III with 1GB of main memory. The machine designated as the router has 2.8GHz Pentium 4 with 1GB of main memory. All machines are connected to a switched Gigabit Ethernet using DLink DGE-550T Gigabit Ethernet cards. We set up a 40ms delay path between the initiator and target using Nist Net[2]. The bottleneck bandwidth is set to 100 Mbps using NistNet. We use file system benchmark *Postmark* to generate storage traffic for our experiments. The general ns-2 simulation setup we use is as shown in Figure 5. For our experiments, the buffer space at the bottleneck router N1 is set to 50 packets and the bottleneck link has a bandwidth of 100 Mbps. The round trip time(RTT) between the endpoints is set to 40ms. The initiator is fed with traces provided by HP Labs. These traces are representative of the workload typically found in file servers. The I/O traces contain information about when the request starts, whether it is a read or a write operation, the request data size, the selected disk, and the sector address. The disk behavior we simulate is that of the Seagate

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

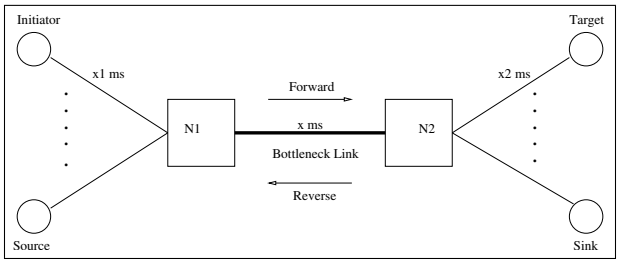

Figure 5. ns-2 Simulation Setup

ST39102LW Cheetah9LP disks. The specifications for this disk can be found at [1].

6. Results

The parameters used to compare the Advanced TCP Stacks are throughput achieved and TCP fairness behavior.

6.1. Effect of the Number of Connections per iSCSI Session

In the first set of experiments, we compare the performance of the various TCP stacks considered with different number of TCP connections comprising the iSCSI Session. The results obtained using the emulation testbed are

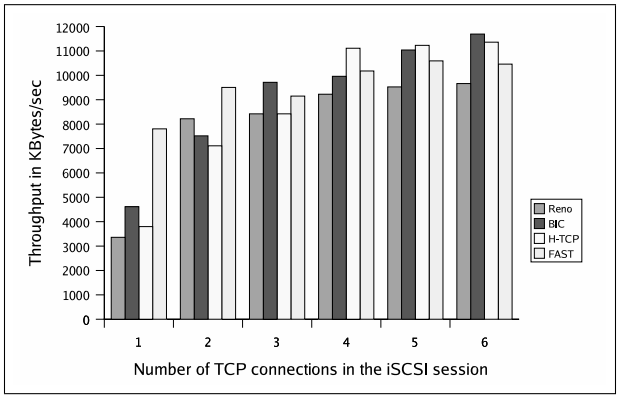

Figure 6. Comparison of TCP Variants for iSCSI data flows

shown in Figure 6. Here we compare TCP Reno with FAST TCP, BIC-TCP and H-TCP. We observe that for less number of connections(less than 3), FAST TCP gives best performance among the variants considered, and BIC TCP and H-TCP perform poorly. As the number of connections increase, both BIC TCP and H-TCP show an improvement in throughput achieved, however rate of improvement in the throughput achieved using FAST TCP diminishes.

This behavior of FAST TCP can be attributed to the presence of reverse traffic. The SCSI data transfers involve both reads and writes. Hence at any given point in time data transfers are taking place in both directions across the network. Thus, for the write transfers directed from the initiator to the target, the read transfers act as the reverse traffic and vice versa. Reverse traffic causes queuing on the reverse path. This in turn can result in the ACKs being lost or coming back in bursts(compressed ACKs). This modification of the ACK behavior results in the sender observing increased RTTs due to increase in the reverse path delays. FAST TCP is based on TCP Vegas and uses the queuing delay in addition to packet loss as the congestion measure. For FAST TCP, an increase in the queuing delay causes the FAST TCP congestion window to change (increase in this case) at a slower rate and the equilibrium point is shifted downwards. Thus the network bandwidth is under utilized. As the number of connections sharing the link increase, the reverse traffic increases resulting in degradation of the performance obtained.

Using simulations, we perform the same experiment. However, since the ns-2 release for FAST TCP is currently not available, we do not consider it in our simulation studies. We compare TCP Reno with BIC–TCP, HSTCP, Scalable TCP and H-TCP. We ran the simulation for 1000 seconds. The results for the simulation are as shown in Figure

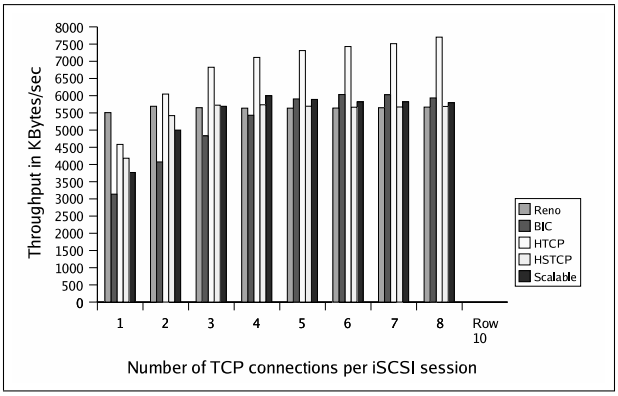

Figure 7. Comparison of TCP Variants using ns-2 simulations

7. We observe that for fewer number of connections, performance obtained using the newer variants is poor compared to that obtained with TCP Reno. Also, H-TCP gives the best performance for most of the observed cases, except for the case of one TCP connection, where TCP Reno outperforms the other TCP variants. For higher number of connections (five or more), identical performance achieved with BIC–TCP and Scalable TCP.

To determine the reasons for the poor performance of the advanced TCP stacks for lesser number of connections, we determine the number of packet drops for BIC TCP, H–TCP and Reno TCP, for the different number of TCP con-

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

nections considered. Both HSTCP and Scalable TCP result in far more packet losses as compared to Reno, BIC–TCP and H–TCP. Hence, we do not plot the losses for HSTCP and Scalable TCP. This is shown in Figure 8. It can be

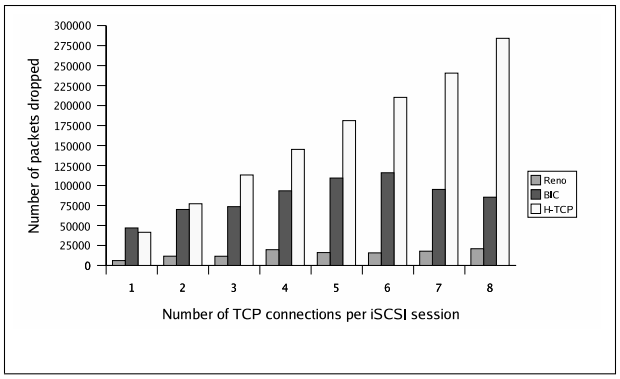

Figure 8. Packet Drops for TCP Variants

observed that the use of BIC TCP and H-TCP algorithms results in far more packet losses as compared to TCP Reno. This can be attributed to the more aggressive window increment strategy used by these algorithms in the congestion avoidance phase. Also, H-TCP is more aggressive as compared to BIC TCP. These losses result in the poor performance observed for lesser number of connections.

6.2. Effect of the iSCSI parameter MaxBurstLength

In the second set of experiments, we studied the performance of the different TCP variants with variation in the iSCSI parameter Maximum Burst Length. The results obtained using the emulation testbed are shown in Figure 9. We use the testbed described in Section 5. The number of connections that comprise the iSCSI session were chosen to be 5, for these experiments. Both FAST

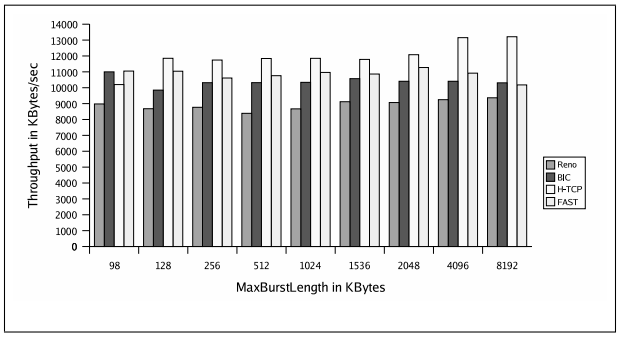

Figure 9. Effect of MaxBurstLength on performance obtained with TCP Variants

TCP and BIC TCP give better performance as compared to TCP Reno across all values of the MaxBurstLength. H-TCP gives higher throughput values for almost all values of the MaxBurstLength considered. The poor performance of TCP Reno is due to its AIMD congestion avoidance behavior. Figure 10 shows the results obtained using ns-2 simu-

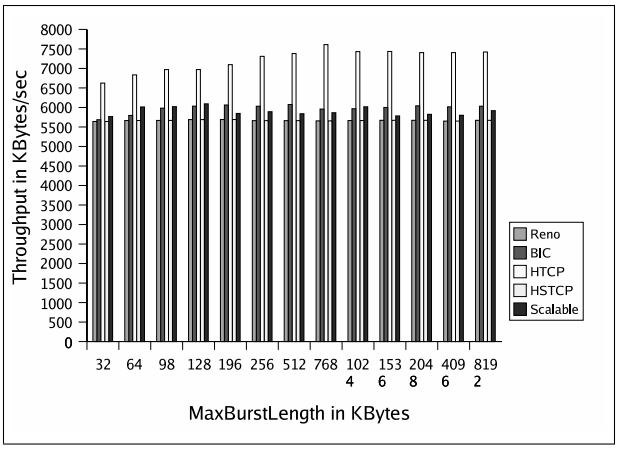

Figure 10. Effect of MaxBurstLength on performance obtained with TCP Variants using ns-2 Simulations

lation. Here, the number of connections is chosen to be 5 as in the experiments conducted using the testbed setup.

The graph indicates that over links with large bandwidth delay products, H-TCP outperforms the other TCP variants considered. Further, the performance achieved using the Reno and HSTCP congestion control algorithms is nearly the same. BIC TCP performs relatively better as compared to Scalable TCP. Also, for MaxBurstLength values greater than 256KB, approximately the same performance is observed across the TCP variants. For lower values of MaxBurstLength, the burst data is not sufficient to utilize the TCP congestion window, and hence we observe lower throughput numbers.

6.3. Fairness

From the results presented in sections 6.1 and 6.2, we observe that H-TCP performs better than the other TCP variants overall. However, as seen in Figure 8, H-TCP results in more packet drops. Also, BIC-TCP results in relatively lesser losses as compared to H-TCP but more losses as compared to Reno TCP. This suggests that H-TCP is more aggressive as compared to the other variants considered. We do not consider HSTCP and Scalable TCP here, as results in [14] indicate that these variants are unfair to other flows. To determine the fairness properties of H-TCP and BIC-TCP, we use our simulation setup. We establish 2 TCP flows sharing the bottleneck link. The two flows

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

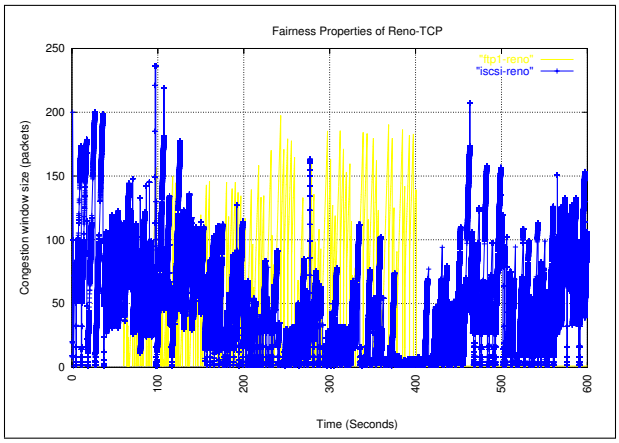

Figure 11. Fairness Properties of TCP Reno using ns-2 simulations

comprise one iSCSI flow starting at 0 seconds and a FTP flow F1 starting 60 seconds. Flow F1 ends at 400 seconds while the iSCSI flow finishes at 600 seconds. Both the flows use the same TCP congestion control algorithm. The

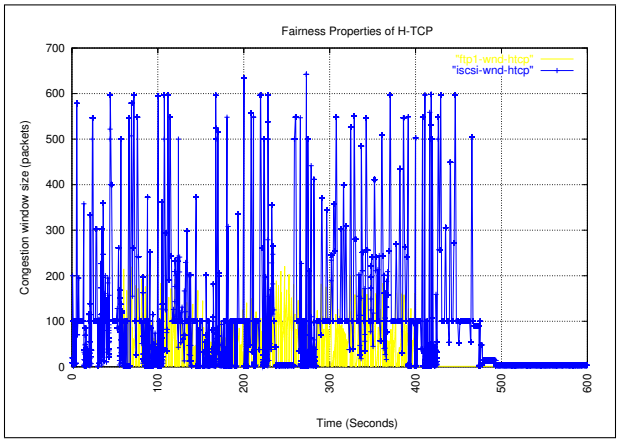

Figure 12. Fairness Properties of H-TCP using ns-2 simulations

performance obtained with the Standard TCP Reno congestion control algorithm is as shown in Figure 11. When the FTP flow arrives, initially the iSCSI flow dominates and consumes a greater portion of the bandwidth. However, beyond 200 seconds, the FTP flow uses up a greater portion of bandwidth as compared to the iSCSI flow. This is the pattern of sharing till the FTP flow ceases to exist at 400 seconds. After 400 seconds, the iSCSI flow increases its window size to use up the newly available share of the bandwidth.

In an identical environment, the results for H-TCP are as shown in Figure 12. It can be observed that the iSCSI flow consumes most of the bandwidth available resulting in an unequal sharing of the bandwidth when the FTP flow F1 arrives. The FTP flow window averages around 60 packets(packet size 1500 bytes) whereas the iSCSI flow consumes an average window of size 120 packets. Thus, H-TCP results in an unfair sharing of the bottleneck bandwidth. The plot obtained with BIC-TCP congestion control

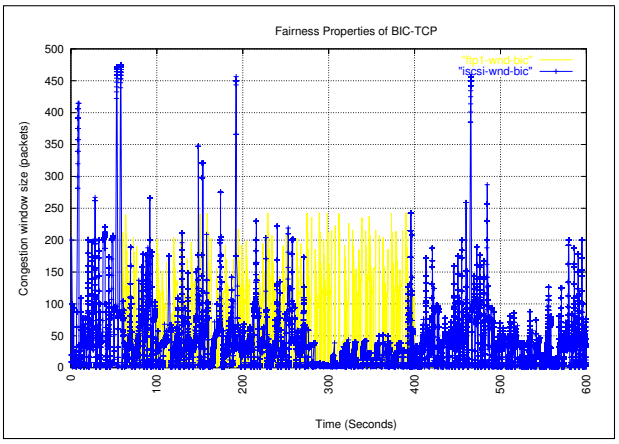

Figure 13. Fairness Properties of BIC-TCP using ns-2 simulations

algorithm is as shown in the Figure 13. With the arrival of the FTP flow F1 at 60 seconds, the two flows have a nearly equal sharing of the available bandwidth with the two flows having alternate periods of large and small window sizes. However, in the period between 275 to 400 seconds, the FTP flow utilizes most of the available bandwidth, with the iSCSI flow window dropping to an average value of 30 packets. A similar drop in the packet sizes for one of the flows occurs when TCP Reno is used to control the flows as shown in Figure 11. When the FTP flow ends, the iSCSI flow takes up the newly available network bandwidth. Thus BIC-TCP is relatively fair as compared to H-TCP, from these results. This can be attributed to the fast convergence mechanism used by BIC-TCP.

7. Related Work

The work in [3], evaluates iSCSI as a competitor to Fibre channel for use in SANs. The work compares performance achieved with four configurations : two using a commercial grade SAN employing Fibre channel as a back end storage medium and iSCSI target hardware, and the other two using iSCSI implementation in software[11]. The experiments to study iSCSI performance using specialized hardware were for a data center environment. These experiments studied the effect of different data block sizes in four configurations. The second set of experiments studied iSCSI perfor-

Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)
0-7695-2318-8/05 $ 20.00 IEEE

mance in a WAN configuration using software based initiator and target implementation with varying network path delays (obtained using a WAN simulation tool) and block sizes. They do not consider variants of TCP algorithm in their work. Our work uses a testbed similar to the one used in the second set of experiments and simulations, software based initiator and target implementations to study the performance of the iSCSI protocol with different TCP stacks, varying number of connections per session and different values of the iSCSI protocol parameter MaxBurstLength.

In [12], the effect of the iSCSI parameter MaxBurstLength on throughput achieved in the presence of congestion is evaluated, which is similar to our work. However, their iSCSI implementation in ns-2 is a basic one, supporting only one connection and does not consider disk behavior. Also they do not consider the different TCP stacks for the performance comparison.

[5] evaluates the performance of the advanced TCP stacks such as FAST TCP[8], BIC-TCP[14], HTCP, Scalable TCP and the standard Reno TCP over high speed production networks using iperf with 3 different path delay configurations. However, their work does not study the performance with storage based protocols such as iSCSI.

8. Conclusions

We have compared the performance of the advanced TCP stacks in an iSCSI environment. We have also studied the effect of the number of TCP connections comprising an iSCSI session and the iSCSI protocol parameter MaxBurstLength on the performance.

Our results indicate that H-TCP outperforms the other TCP variants in throughput performance. However, it results in unfair sharing of the network bandwidth when simultaneous multiple flows exist. FAST TCP is constrained due to the presence of reverse data flows. Throughput Performance obtained using BIC-TCP is second to that obtained using H-TCP and it is relatively fairer to the other flows.

Also, increasing the number of connections for the iSCSI session results in improved performance. The iSCSI parameter, MaxBurstLength does not affect the performance achieved. However the MaxBurstLength should be greater than some minimum threshold, to utilize the TCP Congestion Window.

References

[1] Seagate SCSI Disk Manuals, http://www.seagate.com/support/disc/manuals/.

[2] Nist internetworking technology group. nist net network emulation package, http://www.antd.nist.gov/itg/nistnet/, June 2000.

[3] S. Aiken, D. Grunwald et al. A performance analysis of the iscsi protocol. In *20 th IEEE/11 th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2003.

[4] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall. Improving simulation for network research. Technical Report 99-702, 1999.

[5] H. Bullot, R. Cottrell, and R. Hughes-Jones. Evaluation of advanced tcp stacks on fast long-distance production networks. In *Proceedings of PFLDnet*, 2004.

[6] Sally Floyd. High Speed TCP for Large Congestion Windows, IETF Internet Draft, draft-floydhighspeed-02.txt, February 2003.

[7] G. Ganger, B. Worthington, and Y. Patt. The Disksim Simulation Environment Version 3.0 Reference Manual. Technical Report CMU-CS-03-102, Carnegie Mellon University, January 2003.

[8] C. Jin, D. Wei, and S. H. Low. FAST TCP: Motivation, Architecture, Algorithms and Performance. In *Proceedings of the IEEE INFOCOM 2004*, March 2004.

[9] J. Katcher. Postmark: A new file system benchmark. Technical Report TR3022, Network Appliance Inc., october 1997.

[10] T. Kelly. Scalable tcp: Improving Performance in HighSpeed Wide Area Networks, 2003.

[11] Ashish Palekar, Narendran Ganapathy et al. Design And Implementation Of a Linux SCSI Target For Storage Area Networks. In *Proceedings of the 5th Annual Linux Showcase & Conference*, November 2001.

[12] George Porter, Randy Katz et al. The OASIS Group at U.C. Berkeley: Research Summary and Future Directions. Technical report, UCB, May 2003.

[13] R. Shorten and D. Leith. H-TCP: TCP for High-speed and Long-distance Networks. In *Proceedings of PFLDnet*, 2004.

[14] L. Xu, K.Harfoush, and I. Rhee. Binary Increase Congestion Control(BIC) for Fast, Long-Distance Networks. In *Proceedings of the IEEE INFOCOM 2004*, March 2004.