

A practical approach to TCP high speed WAN data transfers

S. Ravot, Y. Xia, D. Nae, X. Su, H. Newman, J. Bunn
 California Institute of Technology
 1200 E California Blvd, Pasadena CA 91125

O. Martin
 CERN
 1211 Geneva 23, Switzerland

Abstract— In this paper we describe the current state of the art in equipment, software, networks design and methods for transferring large scientific datasets at high speed around the globe, identifying the current limitations and bottlenecks. Based on our practical experience, we first review the limitations of the Transmission Controlled Protocol (TCP) over high speed/latency networks. We explain and illustrate with simple measurements why TCP performs well in a LAN but not so well in a WAN. We follow this with some details on the hardware and software environments, including protocol stack, network interface and system settings which we used to set several Internet2 land speed records over the last couple of years. We then describe current developments that have the goal of transferring High Energy Physics data files from disk servers at CERN¹ in Geneva via a 10Gb/s network path to disk servers at Caltech in California (a total distance of 11,000 km) at a rate exceeding 1 GByte per second. While such transfers represent the bleeding edge of what is possible today, they are expected to be commonplace at the start of the LHC² experiments in 2007.

Index Terms—High performance networking, High speed data transfer, TCP.

I. INTRODUCTION

Physicists are conducting a new round of experiments to probe the fundamental nature of matter and space-time, and to understand the composition and early history of the universe. These experiments face unprecedented engineering challenges due to the volumes and complexity of the data, and the need for collaboration among scientists working around in the world. A new generation of Grid systems is being developed to meet these challenges, and to support the next generation of experiments that are under construction at the CERN laboratory in Geneva. The massive, globally distributed datasets which will be

acquired by these experiments, are expected to grow to the 100 Petabyte level by 2010, and will require data throughputs on the order of gigabits per second between sites located around the globe.

TCP is the most common protocol used for reliable data transfer over IP networks. Since TCP was introduced in 1981[1], network topologies and capacities have evolved dramatically. Although TCP has proved its remarkable capabilities to adapt to vastly different networks, recent theories [5,6] have shown that TCP becomes inefficient when the bandwidth and the latency increase. In particular, TCP's additive increase policy limits its ability to use spare bandwidth.

In this paper we describe experiments that illustrate TCPs limitations. We report on our measurements using the LHCnet, one of the largest network testbeds available today, having 10 Gb/s links connecting CERN in Geneva, Starlight in Chicago and the Caltech campus in Pasadena.

In light of TCP's limitations, we then explain how we have tuned the end-systems and TCP Reno parameters to achieve record breaking data transfers. Finally, we present an experiment currently underway in our group to transfer High Energy Physics data files from a disk server at CERN via a 10Gb/s network path to a disk server at Caltech (a total distance of 11,000 km) at a rate exceeding 1 Giga Byte per second. Whilst such transfers represent the bleeding edge of what is possible today, they are expected to be common practice at the start of LHC experiments in 2007.

II. TCP LIMITATIONS ILLUSTRATED BY EXAMPLES

A. TCP background³

TCP is a reliable data protocol that operates across packet-switched networks. It identifies packets with sequence numbers and uses acknowledgements to allow the sender and the receiver to coordinate with one another to achieve reliable packet transfer. Concurrently, the congestion control mechanism underlying a TCP connection avoids collapses due to congestion and ensures the fairness of network usage.

TCP uses a control variable called the congestion window. The congestion window is the maximum number of unacknowledged packets a source can send, i.e., the

¹ CERN is the world's largest particle physics laboratory. See www.cern.ch

² The Large Hadron Collider (LHC) Project is an accelerator which brings protons and ions into head-on collisions at higher energies than ever achieved before. This will allow scientists to penetrate still further into the structure of matter and recreate the conditions prevailing in the early universe, just after the "Big Bang".

³ This is a brief and simplified description of TCP, a more complete reference is [2].

number of packets in the pipe formed by the links and buffers along a transmission path.

Congestion control is achieved by dynamically adjusting the congestion window according to the additive-increase and multiplicative-decrease algorithm (AIMD). During the congestion avoidance phase, without any packet loss, the congestion window is incremented at a constant rate of one segment per round trip time (additive increase). Each time a loss is detected, the congestion window is halved (multiplicative decrease). Note that in the most widely deployed TCP version (TCP Reno and its variants) the only feedback from the network used to adjust the congestion control algorithm is packet loss.

Due to its elegant design, TCP has achieved remarkable success in efficiently using the available bandwidth, in allocating the bandwidth fairly among users, and – importantly – in reallocating bandwidth shares expeditiously as the use and/or available bandwidth capacity changes over time. However, recent developments have provided evidence that TCP is unable to take advantage of long-haul backbone network capacities in the 10 Gbps range. In the following section we illustrate this problem based on our experience with managing the transatlantic LHC network.

B. Testbed description

The California Institute of Technology and CERN have deployed (in the context of the DataTag[3] project) one of the largest networking testbeds having 10 Gbps access capabilities connecting the Starlight facilities in Chicago and the CERN computing center in Geneva through an OC-192 circuit. The testbed has been extended to the Caltech Campus at Pasadena (CA) through the shared IP backbones of Abilene and CENIC, and a 10 gigabit per second local loop dedicated to R&D traffic between downtown Los Angeles and the Caltech campus in Pasadena. This testbed, shown on Figure 1 is an ideal facility for gathering experimental data on TCP’s performance.

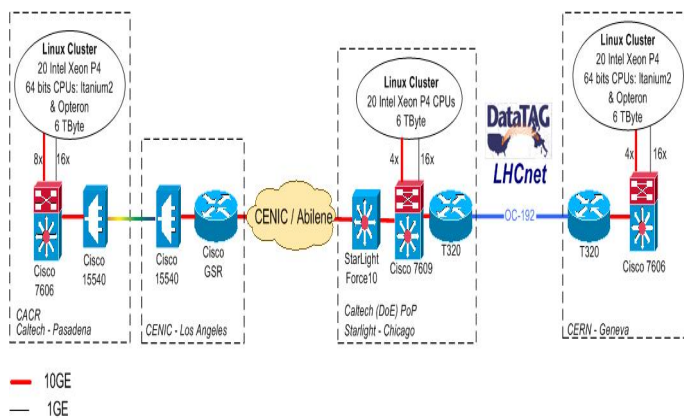


Figure 1: Transatlantic testbed

C. Poor TCP responsive

As administrators of a high speed transatlantic network, we regularly receive complaints about network

performance when distributed applications are able to achieve only a small fraction of the nominal bandwidth.

There are two well-known possible reasons for such poor performance. The first reason is a high transmission error rate. While a packet loss can be caused by congestion, it can also be caused by random bit errors. Since TCP lacks an error-nature classification mechanism, the congestion window is halved unnecessarily when there are packet losses caused by bit errors, even though bandwidth is still available. Our measurements have shown that the loss error rate is zero on our testbed. We are able to transmit data at more than 6 Gbps across our un-congested testbed for several hours without experiencing a single packet loss.

The second common cause of problems is an improper setting of TCP buffer sizes. For example, [4] shows that tuned TCP buffers provide a factor of more than 20x performance gain for connections between Lawrence Berkeley National Lab in California and CERN in Geneva (which has a 180 ms Round Trip Time (RTT)). With tuned TCP buffers, the measured transfer speeds increased by more than an order of magnitude. However, buffer size adjustments are insufficient by themselves to achieve end-to-end Gb/s throughput and saturate our network.

The problem lies in the congestion algorithm itself. AIMD oscillations degrade bandwidth utilization, especially at the bottom of the AIMD saw-tooth. An additive increase by one segment per RTT after a multiplicative decrease is too conservative and substantially underutilizes the capacity of high-speed optical networks. A new parameter which describes the responsiveness of TCP is introduced in [5]. The responsiveness measures how long it takes to recover from a packet loss and eventually return to the original transmission rate (prior to the packet loss), assuming that the congestion window size is equal to the bandwidth-delay product when the packet is lost. Table 2 summarizes the recovery times on our testbed.

Path	Bandwidth	RTT (ms)	MTU (Byte)	Time to recover
Geneva-Los Angeles	1 Gb/s	180	1500	23 min
Geneva-Tokyo	1 Gb/s	300	1500	1 hr 04 min
LAN	10 Gb/s	1	1500	430 ms
Geneva-Chicago	10 Gb/s	120	1500	1 hr 32 min
Geneva-Los Angeles	10 Gb/s	180	1500	3 hr 51 min
Geneva-Los Angeles	10 Gb/s	180	9000	38 min

Table 2 : TCP’s responsiveness on the assumption that the congestion window increases by one MSS each RTT.

Our experiment shows that the time to recover from a single packet loss on the link between Geneva and Chicago is twice the theoretical value calculated above. This is due to the delayed acknowledgement option which is enabled by default in Linux kernels 2.4 and 2.6. At high speed, an acknowledgement is sent by the receiver for every two packets received. At the other end, the sender doesn’t take into account that two packets are acknowledged by only a

single acknowledgement and thus increments the congestion window by only one segment as if only one packet had been sent. The direct consequence is that the congestion window's growth rate is halved.

It is also observed that TCP's responsiveness is improved by larger MTUs⁴. Jumbo frames (9000 Bytes) accelerate the congestion window increase by a factor of six compared to the standard MTU (1500 Bytes). Jumbo frames not only reduce I/O overhead (i.e. CPU load) on end-hosts, they also improve the responsiveness of TCP. Unfortunately, Jumbo frames are not supported by all network equipment and the coexistence of Jumbo and standard MTUs introduces some fairness issues which are described below.

The poor reactivity of TCP has a direct impact on performance in a lossy environment. TCP is much more sensitive to packet loss in a WAN than in a LAN. We used a packet dropper [9] to measure the effect of packet loss in a LAN (RTT= 0.04ms) and in a WAN (Geneva –Chicago: RTT=120ms). Figure 2 reports the bandwidth utilization as a function of the packet loss rate. Both connections have 1 Gb/s of available bandwidth. This illustrates how TCP is much more sensitive to packet losses in a WAN than in a LAN. For example, if the loss rate is equal to 0.01%, e.g. 1 packet lost every 10000 packets transmitted, the bandwidth utilization is almost 100% in a LAN but only 1.2% in a WAN.

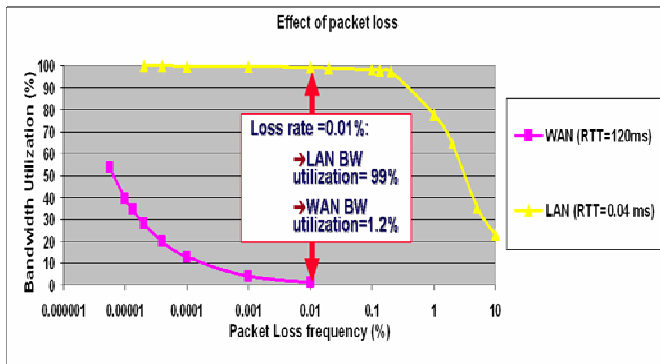


Figure 2: Effect of packet loss over a LAN and a WAN

D. Fairness: TCP (Reno) MTU & RTT bias

Roughly speaking, the fairness of a network can be described as follows: if N identical TCP sessions share the same link, each should get $1/N$ of the link capacity. The AIMD algorithm was designed to converge towards a fair allocation of the bandwidth, but it doesn't take into account the impact of different round trip times and different MTUs. At very high speeds over wide area networks, we have found that different MTUs and different delays lead to a very poor sharing of the bandwidth.

Two TCP flows with different MTU sizes were

⁴ Maximum transmission unit. It defines the largest size of packets that an interface can transmit without needing to fragment.

established between CERN and Starlight on a path that was configured to have a 1 Gb/s capacity limit (Figure 3). The first TCP flow was established with a MTU of 1500 bytes and the second flow with a MTU of 9000 bytes. The results presented in Figure 4 show that for the flow with an MTU of 1500 bytes, the average TCP throughput was 50 Mbps while for the flow with an MTU of 9000 bytes, the average throughput achieved was 698 Mbps. There is a factor of 14 between the two TCP flow speeds!

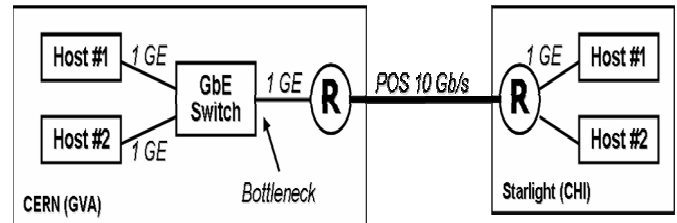


Figure 3: Experiment setup

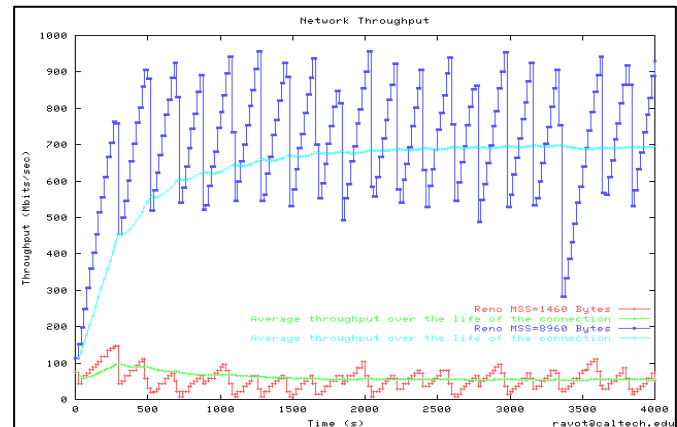


Figure 4: TCP (Reno) MTU Bias

In the second experiment we investigated the effect of delays on fairness. We repeated the same experiment as before, but with the flows having the same 9000 byte MTU but different Round Trip Times. One stream was established between CERN and Starlight, and another between CERN and Caltech, giving RTT measurements of 117ms and 172ms respectively. The results of this test show that the TCP session with the smaller RTT achieved an average throughput of 514 Mb/s compared with the TCP session with the higher RTT that achieved an average throughput of 202 Mb/s.

III. RECORD-BREAKING PERFORMANCE

A. Contest rules

The Internet2 Land Speed Record (LSR) [8] competition for the highest-bandwidth, end-to-end networks is an open and ongoing contest. Internet2 Land Speed Record entries are judged on a combination of how much bandwidth they used and how much distance they covered end-to-end, using standard Internet (TCP/IP) protocols. All the hardware and software used along the path must be publicly available. The contest rules can be found on the LSR

website. Since the year 2000, when the first record entry was filled, the records for the 4 categories (IPv4 and IPv6, single and multi-stream) have been broken several times. The current record shows a factor of 2000 increase over the initial entry.

B. Challenges and limitations

Probably the most restrictive rule of the contest is rule number 3, dictating the use of a standard TCP implementation as described in RFCs 791 and 793. As illustrated in the first part of this paper, the current implementation of TCP has severe limitations when it comes to “Long Fat Pipes”. The limitations due to the congestion avoidance algorithm (AIMD) have a direct implication on high speed tests: **no packets can be lost during the transfer**. A single packet loss would halve the throughput and the time to recover from the loss would destroy the chances of winning the contest.

To avoid this problem, one simply needs to reduce the packet-loss rate! In our environment, packet loss is due exclusively to congestion in the network, i.e., packets are dropped when the number of unacknowledged packets exceeds the available capacity of the network. In order to reduce the packet-loss rate, we must prevent the increase of the congestion window before it reaches a congested state. Because explicit control of the congestion window is not possible, we turn to the flow-control window (TCP buffer sizing) to implicitly cap the congestion-window size to the bandwidth-delay product of the wide-area network so that the network approaches congestion but never actually reaches it.

C. Test Setup

The network path we used crossed dedicated networks (DataTag) as well as shared networks (Abilene/CENIC). We had to take special care to not interfere with production traffic on the shared networks.

A good choice of operating system for the end stations was Linux, due to its being open source. Having access to the source code helped enormously when debugging problems and errors, and also when we came to try improving the performance. We also used machines running Windows OS in collaboration with Microsoft Research. The results are reported on in the following section.

The current record, a memory-to-memory data transfer at 6,5 Gbps with a single TCP stream between Geneva and Los-Angeles, was set using an Opteron (2x Opteron 2.2 GHz Tyan 2882, 2 GB memory) as the sender and an Itanium2 (HP rx4640, 4x 1.5GHz Itanium-2, zx1 chipset, 8GB memory) as the receiver. Both hosts were equipped with S2io⁵ 10 GE network adapters. Each node ran Linux 2.6.6 with Jumbo frames and optimized buffer sizes set to be approximately the bandwidth-delay product. The network topology used to set the single stream record is

shown on Figure 5. All intermediate routers/switches on the path supported 9000 byte MTU.

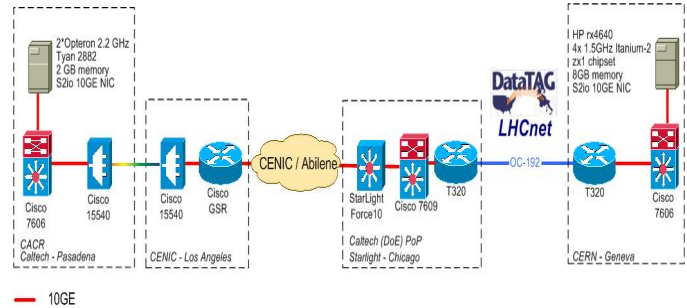


Figure 5: Internet 2 LSR - Single TCP stream at 6.5 Gb/s between CERN and Caltech

The TCP parameters used in all our memory-to-memory data transfer records are described below.

PCI-X burst transfer size. The default PCI-X burst transfer size is 512 bytes, but Intel and S2io 10 GE adapters support a burst size as large as 4096 bytes. We found that increasing the PCI-X burst transfer size to its maximum value provide a 30% throughput increase. We also found that a larger burst size did not provide significant performance improvement for 1500 byte MTU connections. For Jumbo frames, we recommend increasing the PCI-X burst transfer size to its maximum value:

```
setpci -d 17d5:5831 0x62.W=0x2f
```

Transmit and receive queue length. There are two queues between the kernel network subsystems and the network interface card driver. Just as with any queue, an optimal setting is required to avoid buffer overflows because TCP is very sensitive to congestion events. The default sizes of 100 packets for the transmit queue and 300 packets for the receive queue are inadequate for high speed/latency networks which need to absorb temporary bursts. The transmit and receive queue length should be increased⁶ with the following command:

```
/sbin/ifconfig eth0 txqueuelen 30000
/sbin/sysctl -w
sys.net.core.netdev_max_backlog=10000
```

TCP parameter caching. The Linux kernel caches TCP network transfer parameters. In particular, it evaluates the initial slow start threshold from the cached value of the slow start threshold. The idea is to reduce the slow-start probing period by starting the TCP connection with the optimal congestion avoidance settings. In some cases this caching is totally inefficient, especially if the rate of the previous connection was small compared to the rate expected by the new connection. The TCP cache can be flushed with the following command⁷:

⁶ Note that too large queues may affect end-to-end delays.

⁷ Note that the command is not definitive; it has to be executed each time you want to flush the cash.

⁵ www.s2io.com

```
/sbin/sysctl -w sys.net.ipv4.route.flush=1
```

SACKs option. The Selective Acknowledgments (SACK) option is a TCP optimization which can considerably improve performance under certain circumstances. However, at high speed, we found that the Linux TCP SACK implementation suffers from significant performance degradation in the case of a burst of packet losses. Therefore, we recommend disabling the SACK option at high speed:

```
/sbin/sysctl -w net.ipv4.tcp_sack=0
```

TCP buffers. The maximum congestion window size is limited by the amount of buffer space the kernel allocates for each socket. Since a large congestion window is required to reach high throughput, the maximum value for the socket buffer memory has to be adjusted. We recommend setting the buffer size to twice the bandwidth delay product of the path.

```
net.ipv4.tcp_mem = 536870912 536870912
536870912
# sets min/pressure/max TCP buffer space,
default 31744 32256 32768
```

The default value for the buffer sizes can be changed by the application just before opening the socket. We used this feature to limit the maximum size of the congestion window and prevent the increase of the congestion window before congesting the network. In the Iperf tool, the buffer size can be changed with the `-W` option.

In the absence of explicit buffer tuning in the application, Linux will auto-tune the TCP connection by allocating the memory for the TCP buffers.

```
net.ipv4.tcp_rmem = 536870912 536870912
536870912
# sets min/default/max TCP read buffer,
default 4096 87380 174760
```

```
net.ipv4.tcp_wmem = 536870912 536870912
536870912
# sets min/pressure/max TCP write buffer,
default 4096 16384 131072
```

D. LSR History

The LSR competition has helped to establish the feasibility of multi-Gigabit per second single stream IPv4 & IPv6 data transfers. It illustrates that it is possible today, with commercial off-the-shelf components, to achieve transoceanic end-to-end Gb/s throughput across shared IP backbones such as CENIC and Abilene.

Today, the record is regularly cited as a reference in the network community. Its past evolution is useful for future networks planning and design. The achieved performance serves as an excellent benchmark reference to evaluate new TCP implementations (which should be able to reach the same level of performance across uncongested networks).

The history of IPv4 and IPv6 LSRs is shown in Figure 6 and Figure 7. Over the last two years, the IPv4 and IPv6

numbers have increased by a factor 20 and 40 respectively. We note that these rates are much higher than Moore's Law. The increase are evidence that both the end systems' speeds and the network backbone capacities have increased considerably. Even though the performance gap between IPv6 and IPv4 has been reduced by a factor of two in two years, there is still a significant difference between them. This is due to two reasons: (1) Our inability to test IPv6 over a path longer than 11,000 kilometers because a router on the longer path did not support IPv6, and (2) the IPv6 implementation in the Linux kernel 2.6.0 was less efficient than the IPv4 implementation. To illustrate the difference, the end-hosts used to break the IPv6 record, (4 Gb/s between Geneva and Phoenix) could reach 5.64 Gb/s with IPv4 over the same path.

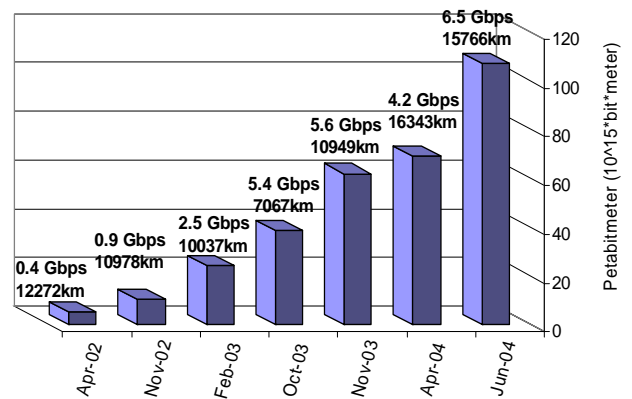


Figure 6: IPv4 LSR History. The Caltech/CERN team did not set the Apr-04 and Apr-02 records. The Jun-04 data has been submitted by Caltech/CERN to the LSR committee and probably qualifies as a new record.

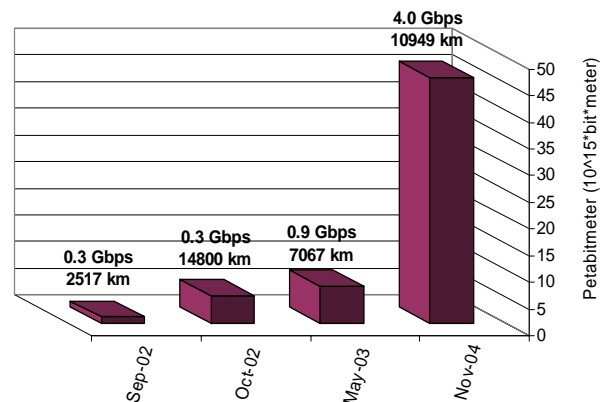


Figure 7: IPv6 LSR History. The Caltech/CERN team set the two most recent records.

Memory-to-memory transfer is only one aspect of high speed data transfers over a Wide Area Network. The next step of practical importance is disk-to-disk transfers. The introduction of storage devices in the setup adds a new degree of complexity to the challenge.

IV. DISK TO DISK TRANSFERS: BREAKING THE 1 GBYTE/S BARRIER

Although memory to memory tests provides significant insights on the performance of the TCP protocol, in practice the transfer of scientific data typically takes place from disk to disk. The memory-to-memory model provides a simplified setup that helps debug many of the network problems, network card driver/operating system problems (e.g. high interrupt usage for network activity), TCP AIMD algorithm problems and so on. Achieving high throughput in memory-to-memory tests is a necessary step towards high speed disk-to-disk transfers, but it does not guarantee it.

There are a number of potential bottlenecks when making host-to-host data transfers. The Wide Area Network has been the bottleneck for many years but this is no longer the case. For example, the average load on the Abilene backbone is no more than 10 % of the available bandwidth, so there is plenty of unused capacity. As described in this paper we have made significant progress in overcoming the limitations of TCP's use in high speed WANs. And new advances in TCP algorithms such as FAST TCP[11], HSTCP[12] or TCP westwood+[13] are succeeding in improving data transport speeds and reliability. The main remaining obstacle to high speed disk to disk transfers is now the storage systems.

A. Hardware limitation

The critical rate-limiting components inside an end-system are illustrated in Figure 8. This shows the complexity of the situation: the end-hosts have to write/read data from disks and transmit/receive data across the network simultaneously. Those two distinct tasks share many of the same host resources (CPU, PCI-X bus, memory, chipset). The performance achievable separately from the host's memory to its disks, and that from memory to memory across the network, do not automatically equate to the same level of performance for real disk to disk transfers across network.

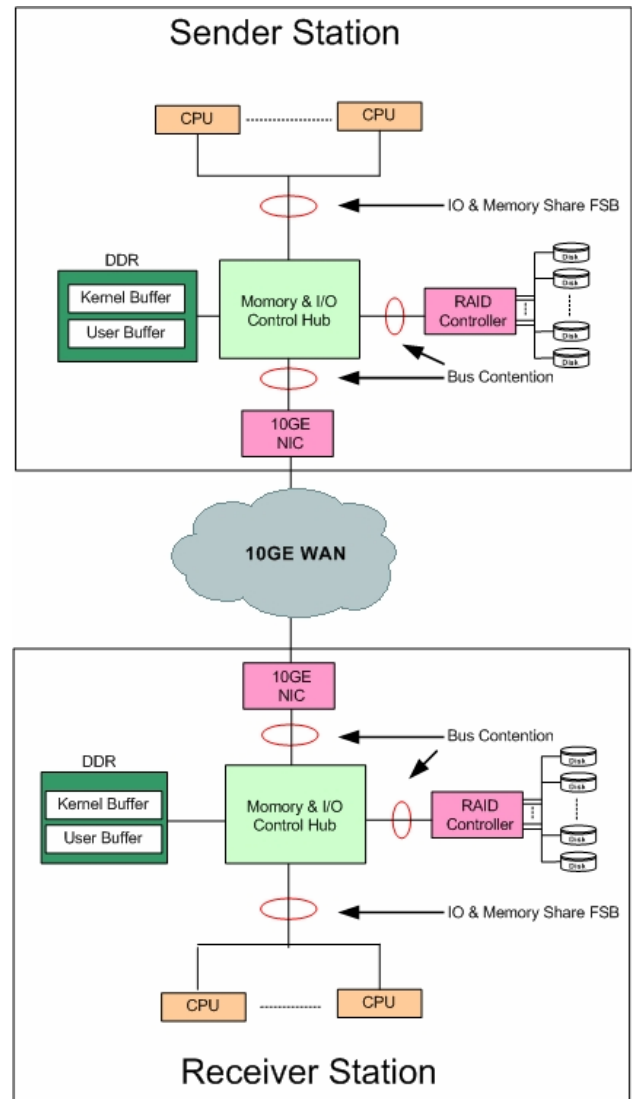


Figure 8: Components that limit data transfer speed

B. PCI-X bus limitation

One of the most important performance limitations for disk to disk transfer comes from the PCI-X bus throughput. The theoretical bandwidth of the 64-bit/133MHz PCI-X bus is 1064MB/s ($64[\text{bits}] \times 133[\text{MHz}]$). However, two important parameters, the PCI-X bus utilization and the PCI-X bus efficiency limit the bandwidth available to data transmission. Without entering into a detailed discussion of the details of the PCI-X overhead we first note that it is not possible to transmit TCP data at 1 GB/s speed across a network adapter inserted in a PCI-X slot. Secondly, if we wish to take full advantage of the 10 GE network adapter, it is not recommended to attach a second PCI-X device to the same bus.

Most currently available systems are equipped with a maximum of only two independent PCI-X buses. If one bus is dedicated to the network adapter, then there is only one left for RAID controllers. In Table 2, we show some results of an evaluation into the efficiency with which two RAID controllers can share the same PCI-X bus. These

measurements show that on two different platforms, the second controller increases the performance linearly.

3ware 7506-8 RAID controller performance under Redhat AS3 On Dual 3 GHz Xeons System		
	Read (MBs/sec)	Write (MBs/s)
2 x Dual 3ware cards + 16 disks	389	323
1 x Dual 3ware cards + 8 disks	194	182

3ware 9506s-8 RAID controller performance under Redhat AS3 on Dual 248 Operons		
	Read (MBs/sec)	Write (MBs/s)
2 x 3ware cards + 12 disks	338	291
1 x 3ware card + 6 disks	264	165

Table 1: Raid controller performance

C. Raid controller performance

We have conducted a set of tests to measure the performance of 3ware RAID controllers and evaluate PATA/SATA disk drive performance. These tests were carried out on four different types of Linux file systems: ext3, jfs, reiserfs and xfs. In these tests, no network transfers were involved since our intention was to establish the performance limits of just the controllers with various file systems. The bonnie++ software⁸ was used to control and measure the sequential memory-to-disk transfers and sequential disk-to-memory transfers.

The first test used a Xeon 3 GHz processor with a Supermicro[14] X5DPE-G2 motherboard (Chipset 7501) equipped with two 3ware 8506-8 controllers inserted in two independent PCI-X buses. We compared SATA vs. PATA disk system performance using Seagate 160GB SATA 150 5400RPM hard drives and PATA WD1200 5400RPM 120GB drives.

For RAID performance, an important tunable parameter of the Linux kernel is the large VM layer read-ahead cache. An appropriate setting can result in more asynchronous reads and yields higher sequential read throughput. The appropriate parameter value in Linux kernels 2.4 and 2.6 can be set with the following commands:

Linux kernel 2.4 tuning parameters:

```
sysctl -w vm.min-readahead=512
sysctl -w vm.max-readahead=512
sysctl -w vm.bdflush= "10 500 0 0 500
3000 10 20 0"
```

Linux kernel 2.6 tuning parameters:

```
blockdev --flushbufs /dev/sda
blockdev --setra 4096 /dev/sda
```

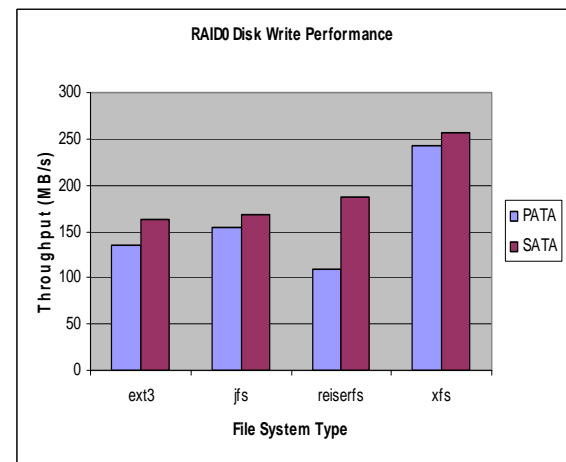
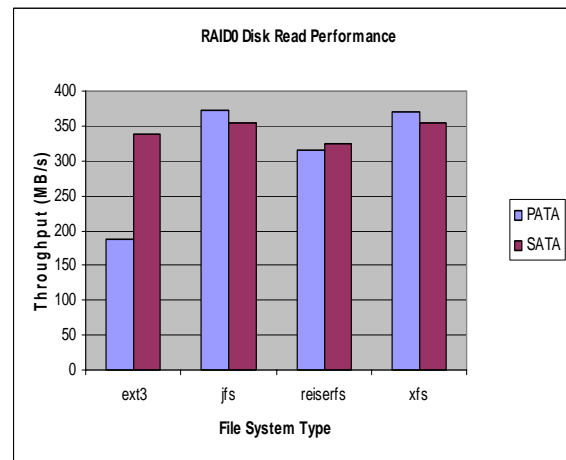


Figure 9: 3ware Raid controllers' performance

The test results are shown in Figure 9. They show that the xfs file system is slightly better in read/write performance compared with the other file systems. (Note that although we tested the old reiserfs, we are planning to test the newer version which is expected to perform better than xfs in some configurations.) In general, the SATA drives perform slightly better (on average 5% better) than PATA drives. This is due to the faster spin speed in the SATA drives used. The SATA drives are easier to install and have better cooling. Another argument against PATA drives is that the cables are often too wide and short making them difficult to work with when space is at a premium.

The second test replaced the 3ware controllers by two Supermicro PCI-X 64-bit/133MHz DAC-SATA-MV8 controllers attached to WD SATA 250GB hard drives. The results are shown in Figure 10: Supermicro controllers achieve disk I/O performance speeds of 250MB/s in sequential read and write of large files using the xfs filesystem. There was no discernible difference in performance between the Supermicro and 3ware controllers.

⁸ The benchmark command line used was `bonnie++ -d /raid -s:9000:64k -mServer -r4096 -x10 -u0 -f -q | bon_csv2txt`. For details about bonnie++ see <http://www.coker.com.au/bonnie++/>

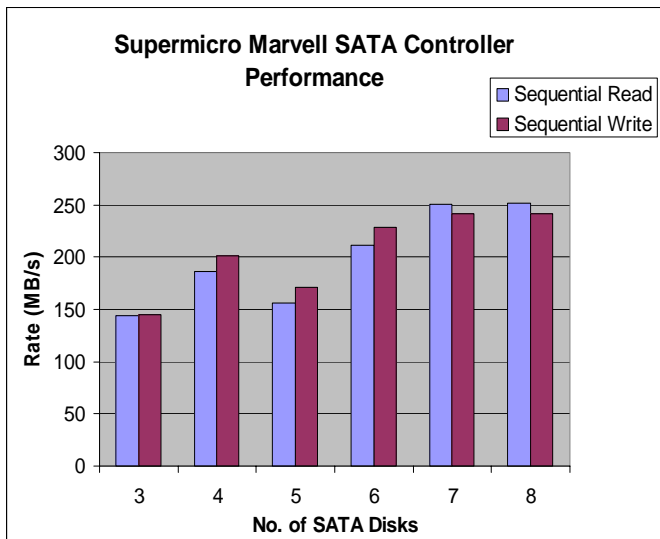


Figure 10: Supermicro DAC-SATA-MV8 controller performance

D. WAN Disk-to-disk

Having established the performance of the RAID subsystems and the memory to memory WAN transfers, the next step is to evaluate disk to disk transfers in the WAN. Currently, the best performance result has been a 200 MByte/s file transfer between CERN and Phoenix. Thus there is a significant performance gap between memory-to-memory and disk-to-disk transfers, basically due to resource sharing bottlenecks in the end systems.

E. Progress in constructing systems and software to break the 1 GB/s barrier

Since April 2004 we have been collaborating with Microsoft, S2io, Newisys and AMD on the design of a prototype server system capable of sustaining 1 GB/s throughput from disk to disk across a long haul WAN. The prototype is based on the Newisys 4300 AMD system⁹, a quad AMD Opteron 848 2.2GHz with three AMD-8131 chipsets and equipped with 16GB of PC3200 DDR memory. Two S2io 10GE NICs are installed in the first and second 64-bit/133MHz PCI slots. Three Supermicro DAC-SATA-MV8 controllers are installed in two 64bit/133MHz and one 64bit/66MHz PCI slots. Each Supermicro controller card has eight Western Digital 250GB SATA 150 7200RPM hard drives in two separated SATA disk cages. The 24 hard drives comprise a single RAID set offering a total capacity of 5TB. The Prototype systems run the 64-Bit Extended Systems Edition for AMD64 of Microsoft's Windows Server 2003.

In order to avoid the 8.5 Gbps theoretical throughput limitation on the PCI-X bus, we use 802.3ad link aggregation to form a logical interface using two physical 10 GE network adapters. This provides a theoretical limit

for the bundled link of 17 Gb/s (twice 8.5 Gb/s) - nominally exceeding our target of 1 GB/s.

Initial back-to-back tests with a pair of these prototypes showed performance figures that were less than expected: 7.86 Gb/s of aggregate throughput from memory-to-memory, with 52.4% of the CPU utilized. To understand and correct this shortfall we are working on optimizing the driver codes, and expect to soon be able to demonstrate 10 Gb/s throughput with reduced CPU load.

The RAID controllers used are Supermicro DAC-SATA-MV8, which we measured with eight drives achieve 445MB/s sequential read and 455MB/s sequential write. This is an 85% increase over the best write performance in the Linux systems, and is mainly due to better drivers and optimization in the Microsoft OS. With three Supermicro controllers and 24 disks, the throughput reaches 1.2GB/s in read and write with a CPU utilization of less than 20%.

Using the same prototype systems, we made tests across the 10Gb/s WAN using a single pair of 10GE network adapters. The maximum transfer speeds measured were:

- memory-to-memory: 4.02Gb/s (8 TCP streams)
- disk-to-memory: 2.2Gb/s (single TCP stream and a 100GB file)
- disk-to-disk: 1.5Gb/s (single TCP stream and a 100GB file)

During these tests, we found that the performance of the Newisys Opteron server as a TCP receiver was lower than the Itanium2 server. Using an Intel Itanium2 as a receiver over 10GE WAN we could reach much higher throughput. This can be attributed to the different way the interrupts are distributed on the two platforms. The results using the Itanium2 server were:

- memory-to-memory: 6.5Gb/s (8 TCP streams)
- memory-to-memory: 6.36Gb/s (single TCP stream)
- disk-to-memory (Newisys to Itanium2): 5Gb/s (single TCP stream and a 100GB file)

V. SUMMARY AND FUTURE WORK

While the current TCP congestion control mechanism has proved to be scalable during the past 20 years, it is less effective on current and next generation ultrahigh speed networks. In this paper we illustrated with practical examples the limitations of the additive-increase multiplicative-decrease (AIMD) algorithm that governs the performance of a TCP connection. We also showed that TCP can be unfair in sharing bandwidth among TCP connections with different delays and MTU sizes.

Despite these limitations, we described how we have established the feasibility of multi-Gigabit per second single stream intercontinental and transoceanic throughput by demonstrating a 6.5 Gb/s transfer using a single TCP stream between Los-Angeles and Geneva. We elaborated on how TCP can be fine tuned to improve its performance effectively on non-congested networks. To cope with realistic large networks where congestion does occur, we

⁹ Newisys 4300 Enterprise-Class Server:
<http://www.newisys.com/products/4300.html>

are working on the development and performance evaluation of new TCP software stacks such as FAST TCP.

The data transport protocol (TCP) is only one component of a complex systems set that together determine the end to end data transfer performance experienced by the user. Other components, such as the end systems' bus architecture, memory subsystem and disk storage configuration and technology all contribute to the achieved data rate. We described how there is a factor of more than ten between memory-to-memory transfer performance and disk-to-disk transfer performance.

We are pursuing a vigorous research and development program in collaboration with partners in the industry with the goal of balancing the systems and software to achieve intercontinental file transfers at rates up to and exceeding 1GB/sec.

ACKNOWLEDGEMENTS

The transatlantic network used for conducting this research is funded by the US Line Consortium (USLIC), which is financed by DoE via Caltech (grant DE-FG03-92-ER40701), NSF (grant ANI 9730202), CERN, Canadian HEP and WHO in Switzerland. The Caltech team is funded by the DoE (grant DE-FC02-01ER25459) and NSF (ANI-0230967). The CERN team is funded by the IST Program of the European Union (grant IST-2001-32459). The authors would like to thank the network operations staff of StarLight, Abilene, NLR and CENIC for their generous support while the testbed was provisioned and the ARTI group at Cisco System for their strong financial support to the deployment of the local loop at Los-Angeles.

REFERENCES

- [1] Jon Postel, "Transmission Control Protocol (TCP) - RFC 793," September 1981.
- [2] W.R Stevens, "TCP/IP Illustrated, Volume 1: The Protocols," Addison-Wesley, 1994.
- [3] "Research & technological development for a Data TransAtlantic Grid," See www.datatag.org.
- [4] T. Dumigan, M. Mathis, B. Tierny, "TCP Tuning Daemon; In Proc. of SuperComputing 2002.
- [5] W. Feng, P. Tinnakornsrisuphap, "The Failure of TCP in High-Performance Computational Grids," Supercomputing 2000.
- [6] S. H. Low, F. Paganini, J. Wang and J. C. Doyle, "Linear Stability of TCP/RED and a Scalable Control," Computer Networks Journal, 43(5):633-647, December 2003.
- [7] J.P. Martin-Flatin and S. Ravot, "TCP Congestion Control in Fast Long-Distance Networks," Technical Report CALT-68-2398, California Institute of Technology, July 2002.
- [8] Internet2 Land Speed Record competition for the highest-bandwidth: <http://lsr.internet2.edu>
- [9] The packet dropper is provided as a patch against the Linux kernel 2.4.20 and is available at https://mgmt.datatag.org/sravot/packet_dropper/
- [10] A description of the problem is available at http://sravot.home.cern.ch/sravot/Networking/TCP_performance/
- [11] Cheng Jin, David X. Wei and Steven H. Low "FAST TCP: motivation, architecture, algorithms, performance", IEEE Infocom, March 2004
- [12] Sally Floyd "HighSpeed TCP for Large Congestion Windows" RFC 3649, Experimental, December 2003.

- [13] L. A. Grieco and S. Mascolo, "A Mathematical Model of Westwood + TCP Congestion Control Algorithm", 18th International Teletraffic Congress 2003 (ITC 2003).
- [14] www.supermicro.com