Survey of Protocols and Mechanisms for Enhanced Transport over LONG FAT PIPES

Abstract

Standard TCP (TCP Reno) is a reliable transport protocol that is well tuned to perform well in traditional networks. However, several experiments and analysis have shown that this protocol is not suitable for bulk data transfer in high bandwidth, large round trip time networks because of its slow start and conservative congestion control mechanism. In this document, we review and compare different emerging alternatives that try to solve this problem in this particular context of very high speed networks. We believe that these innovations are phasing into experimental networks and replacing the stock TCP in high performance networking applications.

Contents

1. Introduction

Existing transport protocols have limitation when they are used in new application domain and for new network technologies. For example, multimedia applications need congestion control but not necessarily ordered reliable delivery. This combination is not offered by TCP [1] or UDP [2]. From another point of view, TCP has been highly tuned with certain assumptions in mind. For example, when a data segment is lost, it assumes that this was most likely due to congestion (i.e. too many segments are contending for network resources) [3]. But, for example, in wireless it could be because of bad reception at the location of the user. So many efforts have been proposed for improving TCP performances in such lossy systems. TCP performance depends upon the product of the transfer rate and the round-trip delay [4]. TCP survived the days of low bandwidth, high latency, and high error rates. But for several reason it is today not able to cope efficiently with the evolving new environment.

In this document, we address the specific problem of transport of bulk data transfers in grid environment (more than 1Gbyte) over high latency, high bandwidth, low loss paths. For a Standard TCP connection with 1500-byte packets and a 100 ms round-trip time, achieving a steady-state throughput of 10 Gbps would require an average congestion window of 83,333 segments, and a packet drop rate of at most one congestion event every 5,000,000,000 packets (or equivalently, at most one congestion event every 1 2/3 hours) [5] . This is primarily due to its congestion avoidance algorithm, based on the Additive Increase Multiplicative Decrease (AIMD) principle. A TCP connection reduces its bandwidth use by half immediately when a loss is detected (multiplicative decrease), but takes 1 2/3 hours to use all the available bandwidth again in this case if no more loss is detected in the mean time. Apparently Standard TCP does not scale well in high bandwidth, large round-trip time networks. A lot of efforts are going on to improve performance for bulk data transfer in such networks. To solve the aforementioned problems, two main approaches are proposed: One focuses on a modification of TCP and specifically the AIMD algorithm, the other proposes the introduction of totally new transport protocols. This is a very active research area in networks.

In very high speed context, performances or loss can also occur due to problems on the host (sender or receiver side). We do not consider these problems in this document. We consider the both cases of shared or not shared links (e.g. dedicated light path).

This draft document attempts to integrate information drawn from sources written for a variety of purposes and which use differing terminology. As a result, this document may be incomplete or contain inaccuracies. Feedback and corrections are welcome. Contributions describing other protocols are also welcome. Direct comments to datatransport-rg@gridforum.org.

2. Methodology and Comparison Criterions

2.1 What are the functions of a transport service and the features of a transport protocol?

Transport layer protocols provide for end to end communication between two or more host. [6] presents a tutorial on transport layer concepts and terminology and a survey of transport layer services and protocols. It classifies the typical services provided by a transport layer. A transport service abstracts a set of functions that is provided to the high layer. A protocol, on the other hand refers to details of how a transport sender and a transport receiver cooperate to provide that service. The following table gives the transport service features:

. CO_message / CO_byte / CL
. No loss / Uncontrolled loss / Controlled loss
. No duplicate / May be duplicate
. Ordered / Unordered / Paritally_ordered
. Data-integrity / No Data- integrity / Partial Data Integrity
. Blocking / Non Blocking
. Multicast / Unicast
. Priority / Np-priority
. Security / No security
. Status reporting / no status reporting
. Quality of service / No quality of service

The following table gives the transport protocol features:

. Connection oriented / Connection less
. Transaction oriented (one request/one response)
. Connection oriented features
- signaling in-band / out of band
- unidirectional / bidirectional
- connection establishment : implicit/ 2 way / 3 way handshake
- connection termination : gracefully / ungracefully

. Error control
- Error detection
- Error reporting
- Error recovery
- Piggybacking
- Cumulative / Selective acknowledgement
- Retransmission strategy
- Duplicate detection
- Forward Error Correction

. Flow / Congestion control
- flow control techniques: sliding window / rate control
- flow control for congestion control: fairness , access control

. Multiplexing / Demultiplexing

. Segementation / Reassembling

2.2 Comparison Criterions

Here we list the criterions we concentrate on when we review and compare these protocols.

- **Performance**

  Standard TCP is not suitable in high bandwidth, large RTT networks because of its low performance in throughput. Therefore, the performance of new protocols should be comparable to TCP in low bandwidth or small RTT networks and much better than TCP in high bandwidth, large RTT networks.

- **Congestion Control**

  Existence of congestion control mechanisms is critical in avoiding congestion collapse. It is important to include reasonable congestion control mechanism if the transport protocol will be used in Internet or other best effort public networks. However, is congestion control still necessary in private networks or quality of service is guaranteed?

- **TCP friendly**

  The term "TCP-friendly" or "TCP-compatible" means that a flow that behaves under congestion like a flow produced by a conformant TCP. A TCP-compatible flow is responsive to congestion notification, and in steady-state uses no more bandwidth than a conformant TCP running under comparable conditions (drop rate, RTT, MTU, etc.). If we strictly abide by this requirement all the time, we will be disappointed again in less congested, large RTT networks. In this document, we only evaluate whether the protocol is TCP friendly in high-congested networks. All protocols in this document are not and should not be TCP friendly in LFP (Long Fat Pipe) networks.

- **Intra-Protocol Fairness**

  There are two kinds of fairness: inter-protocol fairness and intra-protocol fairness. The former is the fairness when the protocol competes with TCP connections. The latter is the fairness among the connections using the protocol. The inter-protocol fairness is the same issue as TCP-compatible. Intra-protocol fairness will be compared among protocols.

- **Easy to deploy**

  When we have plenty amount of bandwidth in underlying networks, what applications need immediately is to deploy transport protocols to utilize the huge bandwidth. In the protocols we are comparing, some need to modify or rebuild operating system kernel, others are just a user level library which applications can call immediately.

- **Predictable**

When we say a protocol is predictable, applications should be able to predict its performance based on current network conditions such as available bandwidth, round-trip time, etc. The purpose is two-fold. Firstly users can tell whether the transport protocol is running correctly by comparing the prediction and actual performance. Secondly protocol developers can systematically identify the factors that influence the overall performance and predict how much benefit any potential enhancement in the protocol might provide. Usually predictability is provided by creating a mathematical analytical model for the protocol.

- **Target Usage Scenario**

"One size fits all" is good but also difficult to accomplish. Before the network speed grew beyond 10 Mbps several years ago, TCP is almost a "One Size fits all" transport protocol. Now it's time to find other solutions for bulk data transfer in LFP networks. These solutions have different preconditions or assumption on underlying networks. Some protocols don't implement congestion control and can only be used in private or QoS-enabled networks. Other seems to be able to coexist with each other and with TCP traffic.

3. Protocol Description

3.1 Comparison between Two approaches

Before going through the detail of each protocols and comparing all of them, we should elaborate the advantages and disadvantages of the two camps.

- **Deployable in Internet**

TCP variants want to take the place of the current standard TCP. In order to be deployable in the Internet or public networks, they create mathematical analytical models and do a lot of simulations to prove the fairness and have not the tendency to cause congestion collapse and therefore adopt sophisticated congestion avoidance algorithms. However, the main motivation of reliable UDP variants is easy to use and good throughput. Usually they are used by a very small amount of users who own a lot of bandwidth. Their applications run on such private networks and seek to utilize the bandwidth as much as possible. They don't intend to substitute TCP in the Internet.

- **Easy to deploy**

Usually reliable UDP variants provide a C or C++ user space library which high performance applications can call. The users don't need to modify or reconfigure the operating system. Instead, TCP variants need patch and rebuild the kernel, which only system administrators can do.

- **Efficiency**

Generally TCP variants are implemented in kernel space whereas reliable UDP variants are implemented in user space. Kernel mechanisms are more scalable and provide better efficiency.

3.2 Reliable UDP Variants

3.2.1 Reliable Blast UDP

Contacts: Eric He (eric@evl.uic.edu), Jason Leigh (spiff@evl.uic.edu)

URLs / RFCs / Papers

- "Reliable Blast UDP : Predictable High Performance Bulk Data Transfer", Eric He, Jason Leigh, Oliver Yu and Thomas A. DeFanti, Proceedings of IEEE Cluster Computing, Chicago, Illinois, September, 2002.
- http://www.evl.uic.edu/cavern/quanta

Principle / Description of Operation

Reliable Blast [7][8] has two goals. The first is to keep the network pipe as full as possible during bulk data transfer. The second goal is to avoid TCP's per-packet interaction so that acknowledgments are not sent per window of transmitted data, but aggregated and delivered at the end of a transmission phase. Figure 1 below illustrates the RBUDP data delivery scheme. In the first data transmission phase (A to B in the figure), RBUDP sends the entire payload at a user-specified sending rate using UDP datagrams. Since UDP is an unreliable protocol, some datagrams may become lost due to congestion or an inability of the receiving host from reading the packets rapidly enough. The receiver therefore must keep a tally of the packets that are received in order to determine which packets must be retransmitted. At the end of the bulk data transmission phase, the sender sends a DONE signal via TCP (C in the figure) so that the receiver knows that no more UDP packets will arrive. The receiver responds by sending an Acknowledgment consisting of a bitmap tally of the received packets (D in the figure). The sender responds by resending the missing packets, and the process repeats itself until no more packets need to be retransmitted.

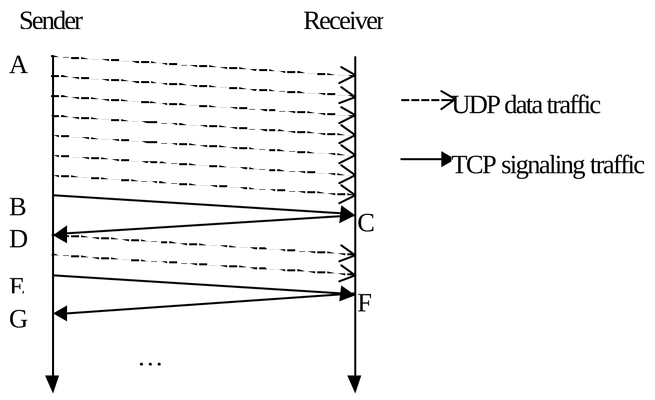

Figure 1. The time sequence diagram of RBUDP

Supported operation mode:
disk-to-disk (i.e. file transfer protocol, not general transport),
memory to memory (general transport)

Authentication: No

Implementations / API: Provides C++ API.

Congestion Control Algorithms:
The congestion control is optional. The algorithm is

```
if (lossRate > 0) {
        $R_{new} = R_{old} * (0.95 – lossRate);$
}
```

R_{new} is updated sending rate after each round of blasting. R_{old} is the sending rate of last round.

Fairness: Not considered.

TCP Friendly: No.

Predictable Performance Model:
The purpose of developing an analytical model for RBUDP is two-fold. Firstly we wanted to develop an equation similar to the "bandwidth * delay product" equation for TCP, to allow us to predict RBUDP performance over a given network. Secondly we wanted to systematically identify the factors that influenced the overall performance of RBUDP so that we can predict how much benefit any potential enhancement in the RBUDP algorithm might provide.

We developed a comprehensive predictable performance model for RBUDP, please refer to our paper on Cluster 2002 for detail.

Results:
We achieve 680Mbps bulk transfer throughput on a 1Gbps link between Chicago and Amsterdam. We think the bottleneck is the memory bandwidth of computers in both sides, especially the receiving side. Please read our paper for the detail result.

Target Usage Scenario:
Bulk data transfer.

Very Aggressive. Only good in private or dedicated networks. The ideal scenario is that you reserve an end-to-end lightpath before running this protocol. You should know how much bandwidth you roughly have. Otherwise you can use iperf or netperf to get the idea.

3.2.2 TSUNAMI

Contacts: Mark Meiss (mmeiss@indiana.edu)

URLs / RFCs / Papers

- Mark R. Meiss "TSUNAMI: A High-Speed Rate-Controlled Protocol for File Transfer
- README file of tsunami-2002-12-02 release. http://www.indiana.edu/~anml/anmlresearch.html

Principle / Description of Operation
The goal of Tsunami [9] is to develop high performance file transfer protocol (running as a user space application) to transfer files faster in high-speed networks than that appears possible with standard implementations of TCP. Tsunami uses UDP for data transfer and TCP for transferring control information. The UDP datagram size is negotiated during the connection setup. The length of the file that is to be transferred is also exchanged during the negotiation phase. Single thread handles both network and disk activity at the sender side whereas the receiver employs separate threads for disk and network activities. Receiver periodically makes retransmission request and retransmissions have higher priority than normal sends. Receiver periodically updates the error rate (calculated based on the number of retransmissions in an interval and previous error rate) to the sender and the sender adjusts its inter-packet delay based on this value. Receiver sends a complete signal after receiving all the datagrams. Tsunami allows the user to configure parameters such as size of the datagram, the threshold error rate (used to adjust sending rate), size of retransmission queue and acknowledgement interval.

Supported operation mode:
disk-to-disk (i.e. file transfer protocol, not general transport)

Authentication: A simple authentication mechanism is used. Upon connection establishment, the server sends a small block of random data to the client. The client xor's this random data with a shared secret, calculates

MD5 checksum, and transmit the result to server. The server performs the same operation and verifies that the results are identical.

Implementations / API:
Implementation available at http://www.indiana.edu/~anml/anmlresearch.html

Congestion Control Algorithms:
Limited. Sending rate is reduced when loss rate is more than the user configurable threshold.

Fairness: Yet to be determined.

TCP Friendly: No

Predictable Performance Model: No

Results:
TSUNAMI recorded mean transmission rate of 850Mb/s for over 17 hours on the Global Terabit research network between Seattle, Washington and Brussels, Belgium. Disk activity was omitted and experiment used a virtual file consisting of short message repeated indefinitely many times. The test that involved actual disk activity was able to achieve 600 Mbps.

Target Usage Scenario:
Designed for faster transfer of large files over high-speed networks.

3.2.3 SABUL (Simple Available Bandwidth Utilization Library)

Contacts:
Yunhong Gu [ygu@cs.uic.edu]
Robert Grossman [grossman@uic.edu]

URLs / RFCs / Papers
http://sourceforge.net/projects/dataspace/

Principle / Description of Operation
The primary goal of SABUL(Simple Available Bandwidth Utilization Library)/UDT (UDP-based Data Transfer Protocol) [10] [11] is to utilize the abundant bandwidth over current long haul networks, such as computational grids. Fairness is important as well. Particularly, two of the major fairness objectives are to be independent of RTT and TCP friendly.

SABUL uses UDP to transfer data and TCP to transfer control information. UDT uses UDP only for both data and control information.

Below is a brief description of data transfer in one direction.

The sender sends out a data packet every inter-packet time, which is updated by the rate control. However, it cannot send out the packet if the number of unacknowledged packets exceeds a threshold, updated by the flow control. A retransmission packet has higher priority than first time packet.

Bit 0: Flag = 1	Bit 1-3: Type	Bit 4-15: Reserved	Bit 16-31: ACK ID or Loss Length
Control Information: type 000 (handshake): maximum window size, MTU type 001 (keep-alive): None type 010 (ACK): acknowledged sequence number, RTT, packet arrival speed, estimated bandwidth type 011 (NAK): loss information type 100 (delay increase warning): None type 110 (ACK^2): None			

Figure 2: Control Packets Structure

The receiver receives and reorders data packet. If it detects packet loss, an NAK packet will be sent back reporting the loss. Selective acknowledgement is used in the protocol, which generates an ACK packet every constant time if there is any packet to acknowledge.

The receiver also measures the packet arrival speed and the link capacity, which will be sent back together with the ACK packet.
The sender sends back an ACK2 packet for each received ACK packet, which is used for the receiver to measure RTT, as well as to decide the next ACK value (i.e., it must be greater than the last received ACK2).

The receiver also checks the RTT variance to check if there is a delay increasing trend. If so, it sends back a delay increasing warning packet to the sender.

Supported operation mode:
memory to memory (general transport)

Authentication:
None.

Implementations / API:
C++ library on Linux/BSD/Solaris.
NS-2 simulation module.

Congestion Control Algorithms:
The SABUL/UDT congestion control algorithm combine rate based control and window based control.

The window based flow control limits the number of unacknowledged packets, and the window size is updated to the product of *receiver side packet arrival speed * (RTT + SYN)*, where SYN is the constant increase interval.

The rate control changes the inter-packet sending time according to packet delay and packet loss. It is using AIMD algorithm, where the decrease factor is 1/9, while the increase factor is related to the link capacity, which is probed using data packet pairs. Note that the increase is not based on any interval related to RTT, but is constant and fixed in SABUL/UDT.

Packet delay increasing trend is detected through the variance of RTT.

The slow start begins with 2-packet flow window and 0 inter-packet interval sending rate. Once an ACK is received, flow window is updated to the number of acknowledged packets until a loss report or a delay increase warning is received, when slow start ends. Sending rate keeps unchanged during slow start phase.

Fairness:
For single bottleneck scenario, the congestion control algorithm is basically AIMD, and it guarantees intra-protocol fairness.

In addition, the fairness is approximately independent to network delay, i.e., connections sharing the same bottleneck but with different RTTs can share the bottleneck bandwidth fairly (equally, in this case).

TCP Friendly:
Yes. Since SABUL/UDT uses delay as sign of congestion, when coexisting with TCP, TCP will occupy most of the bandwidth unless it itself fails to utilize (e.g., because TCP's inefficiency over high BDP network).

Predictable Performance Model:
No. we have not reached a theoretical model to measure the performance accurately. Simulation shows that SABUL/UDT can still reach more than 90% utilization of bandwidth at 1Gbps with 100ms RTT.

Results:
At SC '02, 3 SABUL flows reach 2.8Gbps aggregate throughput between Chicago and Amsterdam.

Target Usage Scenario:
SABUL/UDT is general purpose transport protocol.

3.2.4 Comparison

	RBUDP	**Tsunami**	**SABUL**

Principle	UDP data + TCP control.	UDP data + TCP control.	UDP data + TCP control.
Operation Mode	Disk-to-disk, memory-to-memory.	disk-to-disk (i.e. file transfer protocol, not general transport)	Disk-to-disk, memory-to-memory.
Authentication	No.	Yes but very simple.	No.
FTP syntax	No.	Partially. Only a few commands like connect, get, close, etc.	Partially. Only supports one connection one time.
Application Programming Interface	Yes.	No.	Yes.
3rd Party Transfer	No.	No.	No.
Congestion Control	Optional. Limited congestion control can be turned on.	Limited. Sending rate is reduced when loss rate is more than a threshold.	Yes.
Fairness	N/A	N/A	Yes.
TCP Friendly	No.	No.	Yes.
Predictable Performance Model	Yes.	No.	No.
Implementation	Available.	Available.	Available.

3.3 TCP Variants

3.3.1 HighSpeed TCP

Contacts: Sally Floyd (floyd@icir.org)

URLs / RFCs / Papers:

- "HighSpeed TCP for Large Congestion Windows," Sally Floyd, Internet draft draft-floyd-tcp-highspeed-02.txt, Work in progress, February 2003.
- http://www.icir.org/floyd/hstcp.html

Principle / Description of Operation

HighSpeed TCP [5] aims at improving the loss recovery time of standard TCP by changing standard TCP's AIMD algorithm. This modified algorithm would only take effect with higher congestion windows. i.e,

If congestion window <= threshold, use Standard AIMD algorithm
Else use HighSpeed AIMD algorithm

Standard TCP	HighSpeed TCP
The standard AIMD algorithm is as follows: *on the receipt of an acknowledgement,* w = w + 1/w ; w -> congestion window *and in response to a congestion event,* w = 0.5 * w The increase and decrease parameters of the AIMD algorithm are fixed at 1 and 0.5 With a 1500-byte packets and a 100 ms RTT, achieving a steady state throughput of 10 Gbps would require a packet drop rate at most once every 1 2/3 hours	The modified HighSpeed AIMD algorithm is as follows: *on the receipt of an acknowledgement,* w = w + a(w)/w; higher 'w' gives higher a(w) *and in response to a congestion event*, w = (1-b(w))*w; higher 'w' gives lower b(w) The increase and decrease parameters vary based on the current value of the congestion window. For the same packet size and RTT, a steady state throughput of 10 Gbps can be achieved with a packet drop rate at most once every 12 seconds

Supported operation mode:
Memory to memory (general transport)

Authentication: No

Implementations / API:

- http://www-unix.mcs.anl.gov/~kettimut/hstcp/ HighSpeed TCP implementation for Linux 2.4.19 and initial experimental results from Argonne National Lab.
- http://www.web100.org Tom Dunigan has added HighSpeed TCP to the Linux 2.4.16 Web100 kernel.
- http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/ Experiments of TCP Stack measurements, from SLAC comparing HighSpeed TCP, FAST TCP, Scalable TCP, and stock TCP.
- http://www.hep.man.ac.uk/u/garethf/hstcp/ HighSpeed TCP implementation from Gareth Fairey at Manchester University, for Linux 2.4.19, and initial experimental results with Yee-Ting Li (from UCL).

Congestion Control Algorithms: HighSpeed TCP retains the slow start phase of the standard TCP's congestion control algorithm and the congestion avoidance phase is modified as explained above.

Fairness: The issue of fairness is not explored thoroughly.

TCP Friendly: Unfriendliness increases with decreasing packet drop rates

Predictable Performance Model: The increase and decrease parameters are based on a modified response function. More explanation on the rationale behind the new response function and how it can achieve high throughput with realistic packet loss rates is available in the IETF draft.

Results: Initial experimental results conducted over 100 Mbps link between Argonne National Lab, IL and Lawrence Berkeley National Lab, CA show that HighSpeed TCP performs much better than the standard TCP achieving an improvement of 150%. The round trip time of the connection is 60 ms. The results conducted over 1 Gbps link between Lawrence Berkeley National Lab, CA and Oak Ridge National Lab, TN (with an RTT of 80 ms) show an improvement of 120% for HighSpeed TCP over the standard TCP.

Target Usage Scenario: Initial experimental shows that it performs much better than standard TCP in a dedicated environment. Deployment of this in the broader internet might affect the standard TCP flows.

3.3.2 Scalable TCP

Contacts: Tom Kelly (ctk21@cam.ac.uk) Cambridge University, UK

URLs / RFCs / Papers:

- Tom Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks," First International Workshop on Protocols for Fast Long-Distance Networks, Geneva, February 2003
- http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

Principle / Description of Operation

The main goal of Scalable TCP [12] is to improve the loss recovery time of the standard TCP. The idea is built on the idea of HighSpeed TCP.

Packet loss recovery times for a traditional TCP connection (as well as HighSpeed TCP connection) are proportional to the connection's window size and RTT whereas a Scalable TCP connection's packet loss recovery times are proportional to connection's RTT only.

Slow start phase of the original TCP algorithm is unmodified. The congestion avoidance phase is modified as follows:

For each acknowledgement received in a round trip time,

Traditional TCP	*Scalable TCP*
cwnd = cwnd + 1/cwnd	cwnd = cwnd + 0.01

and on the first detection of congestion in a given round trip time

Traditional TCP	Scalable TCP
cwnd = cwnd – 0.5 * cwnd	cwnd = cwnd – 0.125 * cwnd

Like HighSpeed TCP this has a threshold window size and the modified algorithm is used only when the size of the congestion window is above the threshold window size. Though values of 0.01 and 0.125 are suggested for the increase and decrease parameters, they (as well as threshold window size) can be configured using the proc file system (by a superuser). The default threshold window size is 16 segments.

Supported operation mode:
Memory to memory (general transport)

Authentication: No

Implementations / API:
An implementation for linux kernel 2.4.19 is available at
http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

Congestion Control Algorithms: This work focuses on the congestion control algorithms and proposes a modification for the congestion control algorithm used in the standard TCP.

Fairness: No

TCP Friendly: Claims it does not affect the other standard TCP flows and shows some experimental results involving web traffic to substantiate the claim. More exploration is required before any conclusion can be arrived at this.

Predictable Performance Model: The values of 'a' and 'b' are selected by considering the convergence speed and instantaneous rate variation. The goal was to have faster convergence and smaller instantaneous rate variation.

Results:
The experiments were conducted using a testbed consisting of 12 high performance PCs (6 in Chicago and 6 in CERN, Geneva). The clusters are connected through 2 cisco routers with a 2.4 Gbps link. The PCs are connected to each router through gigabit ethernet ports. The roundtrip time of the connection is 120 ms.

A modified kernel with device transmit queue and receive queue increased to 2000 and 3000 respectively is called gigabit kernel. A significant throughput improvement of 60% to 180% was observed with gigabit kernel compared to standard kernel and a further improvement of 34% to 175% was observed with scalable TCP compared to gigabit kernel Using 16 scalable tcp flows 81% of the maximal performance possible was achieved

Target Usage Scenario:
This is intended to improve the performance of bulk data transport of large data sets with negligible impact on the network traffic. Detailed analysis of the impact on web traffic is yet to be done.

3.3.3 FAST TCP

Contacts: Steven Low (slow@caltech.edu)

URLs / RFCs / Papers:

- "FAST TCP: From Theory to Experiments", C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh; submitted to IEEE Communications Magazine, April 1, 2003
- http://netlab.caltech.edu/FAST/

Principle / Description of Operation

FAST TCP [13] aims to adjust source's sending rate so that link resource is shared fairly by all TCP connections and congestion is avoided with maximum link utilization. Fast TCP totally discards fundamental mechanisms in TCP such as slow start, AIMD and congestion avoidance. Instead, its objective is achieved by implementing two control mechanisms. One is implemented at the source to adjust the send rate dynamically based on an equation and another one is to obtain a congestion measure based on the aggregate flow rate on a link. FAST TCP is similar to TCP in 1) FAST TCP uses same acknowledgement mechanism for reliable delivery; 2) and FAST TCP uses windowing mechanism to control the send rate at the source.

FAST TCP is improved based on a prime-dual model where TCP protocol is modeled by a nonlinear closed-feedback and time-delay control system. Current TCP is not stable when used in a network with high product of capacity and delay. Therefore, to stabilize the send rate at the source, FAST TCP applies an equation to adjust the send rate at the source rather than adopts TCP's AIMD mechanism to control the send rate. The equation for adjusting send rate is obtained by proper parameter assignment and pole-zero placement using Nyquist stability analysis. By properly choosing the equation, FAST TCP can achieve its objective for high performance, stability and fairness in general networks.

FAST TCP is actually a modified version of TCP Vegas. TCP Vegas was introduced as an alternative to the standard TCP (TCP Reno). Vegas does not involve any changes to TCP specification. It is merely an alternative implementation of TCP and all the changes are confined to the sending side. In contrast to the standard TCP, which uses packet loss as the measure of congestion, Vegas source anticipates the onset of congestion by monitoring the difference between the rate it is expecting to see and the rate it is actually realizing. Vegas's strategy is to adjust the source's sending rate in an attempt to keep a small number of packets buffered in the routers along the path. Although experimental results show that Vegas achieves better throughput and fewer losses than standard TCP, Vegas lacked a theoretical explanation of why it works. Here they develop a model of Vegas and show that Vegas can potentially scale to high bandwidth network in stark contrast to the standard TCP. Further, they show that Vegas can become unstable at large delay. Also error in RTT estimation can distort Vegas and can lead to persistent queues and unfair rate allocation. They show that by augmenting Vegas with appropriate Active Queue Management algorithm like Random Exponential Marking (which requires modification in the router), it is possible to avoid the above- mentioned problems. FAST TCP aims at solving those problems by modifying just the TCP kernel at the sending hosts. Detailed description of the algorithm and implementation of FAST TCP is yet to be published (as on 6/1/03, should be out in a couple of months)

Supported operation mode:
Memory to memory (general transport)

Authentication: No

Implementations / API:
Not available yet (as on 6/1/03)

Congestion Control Algorithms:
The congestion control algorithm in FAST TCP is built on the algorithm used in TCP Vegas.

Fairness:
Fair bandwidth allocation is one of the main objectives of FAST TCP but the detail about the mechanism used is yet to be published.

TCP Friendly:
Still under study

Predictable Performance Model:
This work was motivated by their earlier work, which developed a TCP/AQM congestion control system to achieve high utilization, low delay and dynamic stability at the level of fluid-flow models. But the algorithm used in FAST TCP and the theoretical explanation of why it should work is not published yet.

Results:
FAST TCP was demonstrated in a series of experiments conducted during the SuperComputing conference (SC2002). The demonstrations used an OC192 (10Gbps) link between StarLight (Chicago) and Sunnyvale, the DataTAG 2.5 Gbps link between Starlight and CERN (Geneva), an OC192 link connecting the SC2002 show floor in Baltimore and the TeraGrid router in StarLight Chicago and Abilene backbone of Internet2.

Using default device queue size (txqueuelen = 100 packets) at the network interface card and the standard MTU of 1500 bytes, the default Linux TCP (2.4.18), without any tuning on the AIMD parameters, achieved an average throughput of 185 Mbps, averaged over an hour, with a single TCP flow between Sunnyvale in California and CERN in Geneva via StarLight in Chicago with a minimum round trip delay of 180 ms. This is out of a possible maximum of 973 Mbps to the application, excluding TCP/IP overhead, limited by the gigabit Ethernet card, and represents a utilization of just 19%. Under the same experimental conditions, using the default device queue size (txqueuelen = 100 packets) and the standard MTU of 1500 bytes, FAST TCP achieved an average throughput of 925 Mbps (Utilization 95%), averaged over an hour. Even with a device queue size of 10,000 packets, the standard TCP was able to achieve a throughput of only 266 Mbps (Utilization 27%). With 2 TCP flows sharing the path, standard TCP was able to achieve 48% utilization (txqueuelen = 10,000 packets) whereas FAST TCP was able to achieve 92% utilization. With 10 flows, FAST TCP achieved a throughput of 8,609 Mbps (utilization 88%), averaged over a 6-hour period, over a routed path between Sunnyvale and Baltimore, using the standard MTU. The results using the standard Linux TCP implementation for 10 flows are not shown.
In all the experiments described above, the bottleneck was either the gigabit Ethernet card or the transatlantic OC48 link. The experiments conducted using Intel's pre-release experimental 10-gigabit Ethernet card on a single flow from Sunnyvale to Chicago using standard MTU, FAST TCP sustained just 1.3 Gbps. They claim this was due to the limitation in the CPU power at the sending and receiving systems.

Target Usage Scenario: Though it is intended to solve TCP's limitation in high-bandwidth large-delay environments, it is expected to perform well in conventional environments too.

3.3.4 XCP (eXplicit Congestion control Protocol)

Contacts: Dina Katabi (dk@mit.edu)

URLs / RFCs / Papers:

- "Congestion Control for High Bandwidth-Delay Product Networks," Dina Katabi, Mark Handley and Chalrie Rohrs, Proceedings on ACM Sigcomm 2002.
- http://www.ana.lcs.mit.edu/dina/XCP/

Principle / Description of Operation
XCP [14] generalizes the Explicit Congestion Notification (ECN) proposal. Instead of one bit congestion indication used by ECN, it proposes using precise congestion signaling, where the network explicitly tells the sender the state of congestion and how to react to it.

Like TCP, XCP is a window based congestion control protocol intended for best effort traffic. Senders maintain their congestion window (cwnd) and RTT and communicate this to routers via a congestion header (shown in figure 3) in every packet. Sender uses the feedback field in the congestion header to request its desired window increase. Routers monitor the input traffic rates to each of their output queues. Based on the difference between the link bandwidth and its input traffic, router tells the flows sharing that link to increase or decrease their congestion window. It does this by annotating the congestion headers of data packets. Feedback is divided between flows based on their congestion window and RTTs so that the system converges to fairness. A more congested router later in the path can further reduce the feedback in the congestion header by overwriting it. Ultimately the packet will contain the feedback from the bottleneck along the path. When the feedback reaches the receiver, it is returned to the sender in an acknowledgment packet, and the sender updates its cwnd accordingly.

Sender's current cwnd (filled by sender and remains unmodified)
Sender's RTT estimate (filled by sender and remains unmodified)
Feedback (initialized to sender's demands; can be modified by the routers)

Figure 3: Congestion header

Whenever a new acknowledgment arrives, positive feedback increases the sender's cwnd and negative feedback reduces it. An XCP receiver is similar to TCP receiver except when acknowledging a packet it copies the congestion header from the data packet to its acknowledgment.

XCP also introduces the concept of decoupling utilization control from fairness control. A router has both an efficiency controller and fairness controller. The purpose of efficiency controller is to maximize link utilization while minimizing drop rate and persistent queues. It only looks at aggregate traffic and need not care about fairness issues. It computes aggregate feedback at every interval (average RTT of all the flows sharing the link). The aggregate feedback is proportional to both spare bandwidth and persistent queue size. How exactly this aggregate feedback is divided among the packets is the job of the fairness controller. The fairness controller uses the same principle TCP uses (AIMD) to converge to fairness. If the aggregate feedback is positive, allocate it so that the increase in throughput of all flows is the same and if it is negative, allocate it so that the decrease in throughput of a flow is proportional to its current throughput.

Supported operation mode:
Memory to memory (general transport)

Authentication: No

Implementations / API:
XCP implementation in the NS simulator is available at
http://www.ana.lcs.mit.edu/dina/XCP/

Congestion Control Algorithms:
XCP is a congestion control algorithm.

Fairness:
Demonstrates a fairness mechanism and shows how to use it to implement both min-max fairness and the shadow prices model.

TCP Friendly:
Describes a mechanism that allows XCP to compete fairly with TCP but it involves additional work in the routers. Simulation results have been used to demonstrate TCP friendliness of the proposed mechanism.

Predictable Performance Model:
Theoretical analysis on the stability of the protocol and its convergence to fairness can be found in the paper. It is shown to be stable for any link capacity, feedback delay or number of sources.

Results:
The simulations were conducted using the packet level simulator ns-2. The simulations cover link capacities in the range 1.5 Mbps to 4 Gbps, RTTs between 10 ms to 3 sec and number of sources in the range between 1 and 1000. Further, they simulate 2 way traffic and dynamic environments with arrivals and departures of short web like flows. Simulations also show that their results generalize to large and more complex topologies.

They compare XCP with TCP Reno over various Active Queue Management schemes such as RED (Random Early Detection), REM (Random Exponential Marking), AVQ (Adaptive Virtual Queue) and CSFQ (Core Stateless Fair Queuing). The results show that XCP significantly outperforms TCP (with all queuing schemes) in high bandwidth environments as well as in high delay environments. They also show that XCP is efficient in environments with arrivals and departures of short web-like flows. In an environment where the RTTs of the flows that share the bottleneck link vary widely from one another, XCP provides a significantly fairer bandwidth allocation than TCP. They also show how XCP can be used to provide differentiated services to the users based on the price they pay and how XCP can be deployed and how it can gracefully co-exist with TCP.

Target Usage Scenario:
Though XCP is intended to solve TCP’s limitation in high-bandwidth large-delay environments, simulation results show that it performs well in conventional environments too.

3.3.5 CADPC / PTP

Contacts

Michael Welzl, University of Innsbruck
email: michael.welzl@uibk.ac.at web: http://come.to/michael.welzl or http://informatik.uibk.ac.at/users/c70370/

URLs / RFCs / Papers

The PTP website is at http://fullspeed.to/ptp or http://informatik.uibk.ac.at/users/c70370/research/projects/ptp/
CADPC / PTP is mainly the result of a Ph.D. thesis, which is finished and currently in print; a .pdf file is available upon request. A published paper containing some details about CADPC is:

Welzl, M.: "Traceable Congestion Control", ICQT 2002 (International Workshop on Internet Charging and QoS Technologies), Zürich, Switzerland, 16-18 October 2002. Springer LNCS 2511, available from the PTP website.

Principle / Description of Operation

PTP [15], the "Performance Transparency Protocol", is a lightweight signaling protocol that queries routers along a path for performance related information; when used for congestion control purposes , this information consists of:

- the router address
- the MIB2-ifSpeed object (nominal link bandwidth)
- the MIB2-ifOutOctets object (traffic counters)
- a timestamp

At the receiver, it is possible to calculate the bandwidth that was available at the bottleneck during a certain period from two consecutive packets carrying this information for all routers along the path. This operation resembles ATM ABR Explicit Rate Feedback, but work in routers is minimized, all calculations are moved need to network endpoints.

CADPC, "Congestion Avoidance with Distributed Proportional Control", is a congestion control scheme that is solely based on PTP feedback. Since it does not rely on packet loss, it works seamlessly over wireless links. It is slowly responsive in that it utilizes a small amount of feedback, but it shows quick convergence.

Among its outstanding features / properties are:

- good scalability
- fully distributed convergence to max-min-fairness irrespective of RTTs
- designed for heterogeneous links and links with a large bandwidth X delay product
- since it only relies on PTP, it is easily traceable
- simple underlying control law (logistic growth) which is known to be stable
- very smooth rate

Supported operation mode:
memory to memory (general transport)

Authentication: no

Implementations / API: Code is available from the website; future releases will be available from this site, too. Currently, there is:

- A PTP end system and router implementation for Linux
- PTP code for the ns-2 simulator

CADPC was only implemented for the ns-2 simulator so far and will be made available via the PTP website soon.

Congestion Control Algorithms: CADPC is the congestion control algorithm.

Fairness: max-min fairness, could probably be extended to support other forms of fairness (such as proportional fairness) too.

TCP Friendly: No.

Predictable Performance Model: In a network with n users, CADPC converges to 1/(n+1) for each user; thus, the total traffic converges to n/(n+1), which quickly converges to 1 with a growing number of users (these calculations are normalized with the bottleneck capacity). With a VERY small number of users (say, 2 or 3), CADPC is inefficient.

Results:
In simulations, CADPC outperformed several TCP variants and TCP-friendly mechanisms in a large variety of scenarios and in several aspects; in particular, it showed greater throughput than its competitors with close to zero loss. Please see the "traceable congestion control" paper for more details.

Target Usage Scenario:
This protocol is intended for bulk data transport of large data sets. It will work well in scenarios with highly asymmetric links, noisy links and links with a large bandwidth X delay product. It will have trouble if this product is very small. In its present form, it must be isolated from other traffic and will not work well in the presence of short web-like flows or long-term TCP flows. A closer look at CADPC in isolation (usage to control traffic management, or isolated via QoS mechanisms – e.g., by using it within a DiffServ class) is currently under research.

3.3.6 GridFTP

Contact: Bill Allcock (allcock@mcs.anl.gov)

URLs / RFCs / Papers

- “GridFTP: Protocol Extensions to FTP for the Grid”, W. Allcock, J. Bester, J. Bresnahan, S. Meder, S. Tuecke, Global Grid Forum Draft
- “Data Management and Transfer in High Performance Computational Grid Environments”. B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. Parallel Computing Journal, Vol. 28 (5), May 2002, pp. 749-771.
- http://www-fp.globus.org/datagrid/gridftp.html

Principle / Description of Operation
GridFTP [16] is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. It provides a superset of the features offered by various Grid storage systems currently in use by extending the standard FTP protocol. GridFTP builds on RFC 959 (File Transfer Protocol (FTP)), RFC 2228 (FTP Security Extensions), RFC 2389 (Feature negotiation mechanism for FTP), IEFT draft on FTP Extensions.
As defined in the FTP protocol standard it uses two types of channels between the source and the destination namely control channel and data channel. The control channel is used to exchange commands and replies whereas the data channel is used to transfer data. It provides support for parallel data transfer using multiple TCP streams between the source and the destination.
GridFTP also provides support to transfer data that is striped across multiple hosts by using 1 or more TCP streams between m hosts on the sending side and n hosts on the receiving side. GridFTP allows an authenticated third-party to initiate, monitor and control a data transfer between storage servers. Checkpointing is used to provide fault tolerance. A failed transfer is restarted from the last checkpoint. It extends the partial transfer mechanism defined in the standard FTP to support transfers of arbitrary subsets of a file. GridFTP also allows manual or automatic control of TCP buffer size.

Supported operation mode:
File transfer protocol

Authentication:
It implements the authentication mechanisms defined by RFC 2228 (FTP Security Extensions). It supports GSI (Grid Security Infrastructure) and Kerberos authentication with user controlled setting of various levels of data integrity and/or confidentiality.

Implementations / API:
GridFTP is the data management component of the Globus toolkit. It is available at http://www.globus.org/

Congestion Control Algorithms:
As GridFTP uses TCP as the underlying transport mechanism, the congestion control algorithm is same as that of TCP.

Fairness: Same as TCP

TCP Friendly: Yes

Results:
In the experiments demonstrated during SC 2000 between 8 workstations on SC exhibition floor in Dallas, Texas and 8 workstations at Lawrence Berkeley National Lab in California, peak transfer rates of 1.5 Gbps over an interval of 0.1 seconds and 1.03 Gbps over an interval of 5 seconds and a sustained transfer rate of 513 Mbps over 1 hour were achieved. The bottleneck link was a 1.5 Gbps link. The experiments used striped data transfer with 8 striped servers at source and 8 striped servers at destination with a maximum of 4 TCP streams per server. Also, the experimental results show that GridFTP can achieve 78% of the throughput achieved by iperf. The performance difference is attributed to overheads such as authentication, checkpointing etc.

Target Usage Scenario:
Though it is intended for high bandwidth networks, it can be used in conventional environments too.

3.3.8 SCTP (Stream Control Transport Protocol)

Contacts: Pascale Vicat-Blanc Primet (Pascale.Primet@ens-lyon.fr)

URLs / RFCs / Papers

- Ivan Arias Rodriguez: Stream Control Transmission Protocol - The Design of a New Reliable Transport Protocol for IP Networks (this documentation is very detailed and very clear, but long to read...)
- http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb/index.html
- A. Jungmaier, E.P Rathgeb, M. Schopp, M. Tüxen : SCTP - A multi-link end-to-end protocol for IP-based networks, AEÜ - International Journal of Electronics and Communications, 55 (2001) No.1, pp. 46-54 (interesting to have an overview of multi-homing applications)
- RFC 2960: SCTP -- The Stream Control Transmission Protocol, R. Stewart, Q. Xie and al. (the reference for the SCTP protocol)
- http://www.sctp.org/sctpoverview.html (good for beginners...)

Principle / Description of Operation
The Stream Control Transmission Protocol (SCTP) is a new IP transport protocol, existing at an equivalent level as UDP (User Datagram Protocol) and TCP (Transmission Control Protocol), which currently provide transport layer functions to all of the main Internet applications. SCTP has been approved by the IETF as a Proposed Standard, and is currently awaiting allocation of an RFC number.

Originally, SCTP was designed to provide a general-purpose transport protocol for message-oriented applications, as is needed for the transportation of signaling data. It has been designed by the IETF SIGTRAN working group , which has released the SCTP standard draft document (RFC2960) in October 2000.

Unlike TCP, SCTP provides a number of functions that are considered critical for signaling transport, and which at the same time can provide transport benefits to other applications requiring additional performance and reliability. SCTP can be used as the transport protocol for applications where monitoring and detection of loss of session is required. For such applications, some SCTP failure detection mechanisms have been implemented.

The core original features of SCTP are multi-streaming and multi-homing.

Protocol Features:

SCTP is a unicast protocol, and supports data exchange between exactly 2 endpoints, although these may be represented by multiple IP addresses.
SCTP provides reliable transmission, detecting when data is discarded, reordered, duplicated or corrupted, and retransmitting damaged data as necessary.
SCTP transmission is full duplex.
SCTP is message oriented and supports framing of individual message boundaries. In comparison, TCP is stream oriented and does not preserve any implicit structure within a transmitted byte stream.
SCTP is rate adaptive similar to TCP, and will scale back data transfer to the prevailing load conditions in the network. It is designed to behave cooperatively with TCP sessions attempting to use the same bandwidth.

In TCP a stream is referred to as a sequence of bytes, but an SCTP stream represents a sequence of messages (which may be very short or long). The multi-streaming feature allows data to be partitioned into multiple streams that have the property of being delivered independently, so that message loss in any of the streams will only affect delivery within that stream, and not in other streams. In contrast, TCP provides a single stream of data and ensures that delivery of that stream takes place with perfect sequence preservation but causes additional delay when message loss or sequence error occurs within the network. It has been shown that this feature lead too very poor performances on bulk transfer on lossy long bandwidth delay-product links.

The multi-homing is the ability for a single SCTP endpoint to support multiple IP addresses. Using multi-homed SCTP, redundant LANs can be used to reinforce the local access, while various options are possible in the core network to reduce the dependency of failures for different addresses. In its current form, SCTP does not do load-sharing, that is, multi-homing is used for redundancy purposes only.

Supported operation mode: Transport protocol

Authentication: none

Implementations / API: SCTP API

Congestion Control Algorithms: TCP AIMD

Fairness: yes

TCP Friendly: yes.

Predictable Performance Model: yes

Results: The actual SCTP implantation in LINUX seems to offer lower performance than the TCP ones that are generally well optimised (around 10%). SCTP Performance and usage evaluation and algorithm enhancement is actually performed within the INRIA project RESO (http://www.ens-lyon.fr/LIP/RESO/SCTP)

3.3.9 Comparison

	HS TCP	**Scalable TCP**	**FAST TCP**	**XCP**	**CADPC/PTP**	**GridFTP**
Principle	Modify TCP response function when congestion window is larger than a threshold.	More aggressive in congestion control.	Based on TCP vegas.	Decouple efficiency control and fairness control. Former uses MIMD and latter uses AIMD.	Conges tion control based on feedback from routers.	Multiple parallel TCP streams
Operation Mode	Memory-to-memory. Kernel space.	Memory-to-memory. Kernel space.	Memory-to-memory.	Memory-to-memory.	Memory-to-memory.	File transfer protocol
Authentication	No.	No.	No.	No.	No.	GSI and Kerberos
FTP syntax	No.	No.	No.	No.	No.	Yes.
Congestion Control	Yes.	Yes.	Yes.	Yes.	CADPC	Yes.
Need to modify router software	No.	No.	Better if using AQM routers	Yes.	Yes.	No.
Fairness	Need more investigation.	No.	Need more investigation.	Yes.	Max-min fairness	Yes.
TCP Friendly	No when new TCP	Need more investigation.	Need more investigation.	Yes.	No.	Yes.

	response function is triggered.					
Predictable Performance Model	Yes.	Yes.	Need more investigation.	Yes.	Yes.	No.
Simulation and Implementation	Both	Implementation	Not published.	Both.	Both.	Implementation.

4. References

[1] Postel, J. B. Transmission Control Protocol. RFC 793, September 1981.

[2] Postel, J. B. User Datagram Protocol. RFC 768 , September 1980.

[3] Allman, Paxson, et al. TCP Congestion Control, RFC 2581, April 1999.

[4] Jacobson, Braden, et al. TCP Extensions for High Performance, RFC 1323, May 1992.

[5] HighSpeed TCP for Large Congestion Windows, Sally Floyd, Internet draft draft-floyd-tcp-highspeed-02.txt, Work in progress, February 2003.

[6] Iren, S. and Amer, P. The transport layer: tutorial and survey. ACM Computing survey, vol. 31, N°4, December 1999.

[7] E. He, J. Leigh, O. Yu, T.A. DeFanti, “Reliable Blast UDP: Predictable High Performance Bulk Data Transfer,” Proceedings of IEEE Cluster Computing 2002.

[8] E. He, J. Alimohideen, J. Eliason, N. Krishnaprasad, J. Leigh, O. Yu, T. A. DeFanti, “QUANTA: A Toolkit for High Performance Data Delivery over Photonic Networks,” to appear in Future Generation Computer Systems, Elsevier Science Press.

[9] README file of tsunami-2002-12-02 release. http://www.indiana.edu/ ~anml/anmlresearch.html

[10] Yuhong Gu, Xinwei Hong, Marco Mazzucco, and Robert L. Grossman, SABUL: A High Performance Data Transport Protocol, 2002, submitted for publication.

[11] Yuhong Gu, Xinwei Hong, Marco Mazzucco, and Robert L. Grossman, Rate Based Congestion Control over High Bandwidth/Delay Links, 2002, submitted for publication.

[12] Tom Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks," First International Workshop on Protocols for Fast Long Distance Networks, Geneva, February 2003

[13] FAST TCP: From Theory to Experiments, C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh; submitted to IEEE Communications Magazine, April 1, 2003

[14] Congestion Control for High Bandwidth-Delay Product Networks, Dina Katabi, Mark Handley and Chalrie Rohrs, Proceedings on ACM Sigcomm 2002.

[15] Welzl, M.: "Traceable Congestion Control", ICQT 2002 (International Workshop on Internet Charging and QoS Technologies), Zürich, Switzerland, 16-18 October 2002. Springer LNCS 2511, available from the PTP website.

[16] Data Management and Transfer in High-Performance Computational Grid Environments. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Parallel Computing, 2001.

5. Author Information

Eric He
Electronic Visualization Laboratory, University of Illinois at Chicago.
eric@evl.uic.edu

Rajkumar Kettimuthu
Argonne National Laboratory
kettimut@mcs.anl.gov

Yunhong Gu
Laboratory for Advanced Computing, University of Illinois at Chicago
gu@lac.uic.edu

Sanjay Hegde
Argonne National Laboratory
hegdesan@mcs.anl.gov

Michael Welzl
University of Innsbruck, Austria
michael.welzl@uibk.ac.at

Pascale Vicat-Blanc Primet
INRIA, France
Pascale.Primet@ens-lyon.fr

Jason Leigh
Electronic Visualization Laboratory, University of Illinois at Chicago.
spiff@evl.uic.edu

Chaoyue Xiong
Electronic Visualization Laboratory, University of Illinois at Chicago.
cxiong1@uic.edu