GWD-I
Data Transport Research Group

Mathieu Goutelle, ENS Lyon, France
Yunhong Gu, University of Illinois at Chicago
Eric He, University of Illinois at Chicago (editor)
Sanjay Hegde, Argonne National Laboratory
Rajikumar Kettimuthu, Argonne National Laboratory
Jason Leigh, University of Illinois at Chicago
Pascale Vicat-Blanc Primet, INRIA, France (editor)
Michael Welzl, University of Innsbruck, Austria (editor)
Chaoyue Xiong, University of Illinios at Chicago
Muhammad Murtaza Yousaf, University of Innsbruck, Austria
May 10, 2005

A Survey of Transport Protocols other than Standard TCP

Abstract

Standard TCP (TCP Reno) is a reliable transport protocol that is designed to perform well in traditional networks. However, several experiments and analyses have shown that this protocol is not suitable for each and every kind of application and environment – e.g., bulk data transfer in high bandwidth, large round trip time networks. In this document, we review and compare different emerging alternatives that try to solve this and other problems.

Status of This Memo

This memo provides information to the Grid community regarding efficient data transmission. It does not define any standards or technical recommendations. Distribution is unlimited.

Contents

1. Introduction

Existing transport protocols have limitations when they are used in new application domains and for new network technologies. For example, multimedia applications need congestion control but not necessarily ordered and reliable delivery – this combination is not offered by TCP [1] or UDP [2]. TCP has also been designed 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 networks it could be because of bad reception at the location of the user. A large number of proposals have been made to improve TCP performances in such lossy systems. TCP survived the days of low bandwidth and low latency, but for several reasons it is not able to efficiently cope with today's evolving environment.

TCP performance depends upon the product of the transfer rate and the round-trip delay [4], which can lead to inefficient link utilization when this value is very high – as in the case 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 billion 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 meantime.

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[1].

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

Protocol name	TCP variant	UDP based	TCP-friendly congestion control	Router support based	Others
HS-TCP	x				
Scalable TCP	x				
FAST TCP	x				
BIC & CUBIC	x				
Westwood+	x				
UDP lite		x			
RB UDP		x			
TSUNAMI		x			
UDT		x			

[1] See http://www.ens-lyon.fr/LIP/RESO/pfldnet2005

RAP			x		
TFRC			x		
TEAR			x		
LDA+			x		
XCP				x	
CADPC/PTP				x	
SCTP					x
DCCP					x
GRID FTP					x

Table of protocols reviewed

In this document, we review and compare different emerging alternatives that try to solve this and other problems. The table, above, gives a list of protocols we reviewed.

This document attempts to integrate information drawn from sources written for a variety of purposes using 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 Transport service functions and protocol features?

Transport layer protocols provide for end-to-end communication between two or more hosts. Reference [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 is a list of common transport service features:

CO_message	vs	CO_byte	vs	CL[2]
No loss	vs	Uncontrolled loss	vs	Controlled loss
No duplicate	vs	May be duplicate		

[2] CO: connection oriented; CL: connection less.

Ordered	vs	Unordered	vs	Paritally_ordered
Data-integrity	vs	No Data- integrity	vs	Partial Data Integrity
Blocking	vs	Non Blocking		
Multicast	vs	Unicast		
Priority	vs	No-priority		
Security	vs	No security		
Status reporting	vs	no status reporting		
Quality of service	vs	No quality of service		

Service features of TCP are: CO_byte, No loss, No duplicate, Ordered, Data-integrity, Unicast, No Priority and No Security.

The following table is a list of common transport protocol features:

Connection oriented (CO) / Connection less (CL);

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;

Segmentation / Reassembling.

The TCP protocol features are: CO, no transaction oriented, in-band signaling, bidirectional, 3-way handshake, graceful termination, Error detection (sequence number, checksum on header and data) and error recovery (positive acknowledgement with retransmission), window based congestion and flow control. It handles multiplexing / demultiplexing and segmentation / reassembly.

Standard TCP is not very efficient in both interactive communications and in high bandwidth, large RTT networks, due to some of the previous features [7]. Some of the protocols presented here try to relax some of these constraints and do better than TCP in such cases by choosing other service or protocol features.

2.2 Comparison Criterions

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

- *Performance:*

  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 is to be used in the Internet or other best effort public networks. However, is congestion control necessary in private networks, where quality of service may be guaranteed?

- *TCP friendliness:*

  The term “TCP-friendly” or "TCP-compatible" means that a flow 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 may be disappointed again in less congested, large RTT networks. Not all protocols in this document are TCP friendly, in particular 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-compatibility. Intra-protocol fairness will be compared among protocols.

- *Deployment considerations:*

  When we have plenty 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 the operating system kernel, others are just a user level library which applications can call immediately.

- *Analytical model:*

  The purpose of an analytical model is two-fold. Firstly, it enables users to tell whether the transport protocol is running correctly by comparing the prediction and actual performance. Secondly, it enables protocol developers to systematically identify the factors that influence the overall performance and predict how much benefit any potential enhancement in the protocol might provide.

- *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 was almost a “One Size fits all” transport protocol. Now it is time to find other solutions for bulk data transfer in LFP networks. These solutions have different preconditions or assumption on underlying networks. Some protocols do not implement congestion control and can therefore only be used in private or QoS-enabled networks. Other seem to be able to coexist with each other and with TCP traffic.

3. Protocol Descriptions

3.1 TCP Variants

3.1.1 HighSpeed TCP

Contacts:

Sally Floyd (floyd@icir.org)

Standardization Status:

RFC 3649 [5], status: Experimental.

Principle / Description of Operation:

HighSpeed TCP 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 the congestion window is smaller than a given threshold, it uses the Standard AIMD algorithm, else it uses HighSpeed AIMD algorithm.

Standard TCP	HighSpeed TCP
The standard AIMD algorithm is as follows: *upon receipt of an acknowledgement,* w = w + 1/w ; w -> congestion window *and in response to a congestion event*, w = 0.5 * w	The modified HighSpeed AIMD algorithm is as follows: *upon 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 of the AIMD algorithm are fixed at 1 and 0.5.	The increase and decrease parameters vary based on the current value of the congestion window.
With 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	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)

Security features:

None

Implementation / 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.

Usable in the public Internet (TCP-Friendly):

HighSpeed TCP is TCP-friendly when the packet loss ratio is high; its unfriendliness increases with decreasing packet drop rates.

Analytical 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.

Benchmarking:

E. Souza and D.A. Agarwal, A HighSpeed TCP Study: Characteristics and Deployment Issues. LBNL Technical Report LBNL-53215.

HighSpeed TCP performs well in high-speed long-distance links. It falls back to standard TCP behaviour if the loss ratio is high. In case of bursty traffic its link utilization is improved but at the same time there are some issues regarding fairness. The impact of queue management techniques is rather negligible.

Target Usage Scenario:

Initial experiments show 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.

Additional References:

http://www.icir.org/floyd/hstcp.html

3.1.2 Scalable TCP

Contacts:

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

Standardization Status:

Merged with HighSpeed TCP in RFC 3649 [5], status: Experimental.

Principle / Description of Operation:

The main goal of Scalable TCP 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. The 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 upon 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 the 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)

Security features:

None

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 is a modification of the standard TCP congestion control algorithm; it was described as one particular implementation possibility in RFC 3649, thereby integrating it with HighSpeed TCP.

Fairness:

Yet to be determined.

Usable in the public Internet (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.

Analytical 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.

Benchmarking:

Tom Kelly, Scalable TCP: Improving Performance in Highspeed Wide Area Networks, February 2003. URL http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

On a testbed, DataTAG, consisting of 12 high performance PCs with 6 servers located at CERN, Geneva, and 6 servers at StarLight, Chicago, where the PCs were connected to each router through Gigabit Ethernet, Scalable TCP showed a significant throughput improvement when each receiver in Chicago requested a file of size 2 Gigabytes from its associated server in Geneva.

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. A detailed analysis of the impact on web traffic is yet to be done.

Additional references:

Reference [13] and http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

3.1.3 FAST TCP

Contacts:

Steven Low (slow@caltech.edu)

Standardization Status:

IETF Draft: FAST TCP for high-speed long-distance networks Cheng Jin, David X. Wei and Steven H. Low; draft-jwl-tcp-fast-01.txt, June 30, 2003.

Principle / Description of Operation:

FAST TCP is an alternative congestion control algorithm built on TCP Vegas. FAST aims at providing flow level properties such as stable equilibrium, well-defined fairness, high throughput and link utilization. It requires only sender side modification and no co-operation from routers/receivers. FAST TCP uses queuing delay (multi-bit congestion signal) in addition to packet loss to assess and address the congestion where as standard TCP uses packet loss (binary signal) only. Current TCP is not stable when used in a network with a high bandwidth delay product. Therefore, to stabilize the sending rate at the source, FAST TCP applies an equation-

based approach at the source rather than adopt TCP's AIMD mechanism to control the sending rate. By properly choosing the equation and feedback mechanism, FAST eliminates the packet level oscillations, improves the flow level dynamics and achieves its objective of high performance, stability and fairness in general networks.

Supported operation mode:

Memory to memory (General transport)

Security features:

None

Implementations / API:

FAST TCP is divided in two basic layers. The top layer implements FAST functions and algorithms that in principle should be device and operating system independent. One should be able to port the top layer to different systems without modification. The bottom layer implements interface functions that are specific to an operating system or a network device (OS dependent). In addition, few macros need to be inserted into standard kernel code to call various FAST related functions.

Implementation for Linux kernel 2.4.22 and 2.4.24 are available for field trials. Reference: the Netlab website – http://www.netlab.caltech.edu

Congestion Control Algorithms:

The window calculation is divided into three sections: slow start (SS), multiplicative increase (MI), and exponential convergence (EC). SS is essentially identical to the standard slow start in TCP Reno, the only difference being that FAST exits SS when the number of packets queued in the network exceeds a threshold gamma rather than using packet loss. MI is used to rapidly move a FAST connection close to equilibrium whenever it falls below equilibrium. FAST implements a safeguard mechanism in both MI and EC, where a window is increased or decreased on alternative RTTs. In EC, the window moves half-way between the current value and the target in each update interval so it is exponentially increasing, with a negative exponent so the window converges to the target, with time measured in multiples of 10 ms. FAST updates the window based on the following algorithm:

w_new = 1/2{ [w_old *baseRTT/avgRTT] + alpha + w_current

where w_current, w_old and w_new denotes the current congestion window size, the congestion window size one RTT ago and the new target congestion window. alpha specifies the total number of packets a single FAST connection tries to maintain in its path.

Fairness:

FAST TCP uses a log utility function to achieve fairness:

U(x) = alpha * log(x)

Hence, networks with only FAST TCP flows achieve proportional fairness under no congestion or in mildly congested situations (when packet loss occurs infrequently). In severely congested situations, packet loss becomes the dominant signal, and AIMD in FAST should provide comparable fairness as current TCP with identical flows.

Usable in the public Internet (TCP-Friendly):

The parameter alpha can be used to tune the aggressiveness and equilibrium bandwidth share for each FAST connection. Friendliness of FAST TCP to other TCP flows can be tunes by appropriately selecting the alpha value. Also, in low speed networks, when congestion window is small FAST TCP behave exactly like current TCP.

Analytical Model:

This work was motivated by earlier work, which developed a TCP/AQM congestion control (primal-dual) system to achieve high utilization, low delay and dynamic stability at the level of fluid-flow models. The TCP/AQM algorithm is a distributed and decentralized feedback control system where TCP adapts source rates to congestion and AQM feeds back the congestion measures (e.g., loss probability at a link increases or decreases as sources traversing that link increase or decrease their rates). They cooperate to iteratively determine the network operating point that maximizes aggregate utility. When this iterative process converges, the equilibrium source rates are optimal solutions of the primal problem and the equilibrium congestion measures are optimal solutions of the dual problem. The throughput and fairness of the network are thus determined by the TCP algorithm and the associated utility function, whereas utilization, loss and delay are determined by the AQM algorithm.

Benchmarking:

http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/

http://www-iepm.slac.stanford.edu/monitoring/bulk/sc2003/hiperf.html

http://netlab.caltech.edu/results/

In [14], experimental results of first Linux prototype have been presented. Algorithms have been evaluated in both static and dynamic environments in terms of end-to-end throughput and queue behavior in the network. FAST TCP performed well in terms of throughput, fairness, stability, and responsiveness but stability analysis was limited to a single link with heterogeneous sources and feedback delay was ignored. Further, many experimental scenarios were designed to judge the properties of FAST TCP but these scenarios are not very realistic.

Target Usage Scenario:

While it is intended to solve TCP's limitation in high-bandwidth large-delay environments, FAST TCP showed the same performance as current TCP under conventional environments too.

3.1.4 BIC and CUBIC

Contacts:

Injong Rhee
Department of Computer Science, North Carolina State University
rhee AT ncsu dot edu
http://www.csc.ncsu.edu/faculty/rhee/

Standardization Status:

None

Principle / Description of Operation:

BIC is a TCP variant to improve performance in high-speed networks. A unique feature of BIC is its decoupling of scalability and fairness. The prime feature of BIC is its unique window growth function which gives TCP friendliness, enhances the stability of the protocol, improves the network utilization, and makes it fair with other high-speed flows running with same or different round-trip delays. In BIC, congestion control is viewed as a search problem where congestion control needs to find the fair and efficient transmission rate (or "target rate") for the end-to-end path where the current connection is running on. The technique of binary search – a commonly used technique to search for an item in database is used. This binary search technique allows the window increase to be logarithmic; it increases faster when the target rate is far away from the current transmission rate, but slows down as the current transmission rate gets closer to the target. This feature ensures the stability of control while allowing the protocol to become friendlier to existing TCP traffic.

This binary search technique is combined with additive increase (but with a larger increment than the current TCP increment). The additive increase also allows the protocol to increase its window (or rate) faster when there are a lot of available bandwidth.

However BIC can be too aggressive for TCP in short RTT or low speed networks. The growing function of BIC after the window reduction is linear and then logarithmic. When it reaches a maximum value it enters into a new phase called "max probing", where the growth function is exponential and linear. BIC reduces its window by a multiplicative factor after a packet loss.

CUBIC improves the functionality of BIC in terms of having a function, which simplifies window control and enhances TCP friendliness, keeping the major strengths of BIC (particularly stability, and scalability). Since there is only one function, so it simplifies the window control. As the name suggests, CUBIC has a new cubic growth function. In CUBIC, after a window reduction, it grows quickly and when it gets closer to W_{max} (a dynamically calculated maximum window) it slows down and its increment becomes zero at W_{max} . After that, it grows in a reverse fashion, which means that it grows slowly at first and when it moves away from W_{max} , it grows quickly. The slow growth around W_{max} enhances its stability and the utilization of network, and fast growth away from W_{max} makes it more scalable.

Supported operation mode:

Memory to memory (General transport)

Security features:

None

Implementations / API:

ns simulation code for CUBIC can be found at
http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/cubic-script/script.htm

BIC is implemented in the standard LINUX kernel since the 2.6.7 version. A TCP BIC has been made available for Web100.

Congestion Control Algorithms:

The congestion window of CUBIC is determined by the following function:

$$W_{cubic} = C(t - K)^3 + W_{max}$$

where C is a scaling factor, t is the elapsed time since the last window reduction, W_{max} is the window size just before the last window reduction, and

$$K = \sqrt[3]{W_{\max}\beta / C}$$

where β is a multiplicative decrease factor after a packet loss event.

Fairness:

According to [42] the cubic function ensures intra-protocol fairness among the competing flows of the same protocol.

Usable in the public Internet (TCP-Friendly):

When the window size is small, then K is also small, which makes the growth rate of CUBIC function slow. Hence CUBIC becomes TCP friendly when the BDP of the network is small.

Analytical Model:

Analyses of CUBIC in [42] are derived from its congestion window update function.

Target Usage Scenario:

CUBIC is designed for High-speed network environments where TCP cannot fully utilize the network because of its slow growth. At the same time it is equally good for small BDP network, as it becomes very TCP friendly in such an environment.

Benchmarking:

Injong Rhee, and Lisong Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", Third International Workshop on Protocols for Fast Long-Distance Networks, Feb 3,4 2005

Experimental results show that the CUBIC provides good stability and scalability and it is TCP-friendly with both short and long RTTs. Much like HighSpeed TCP, both BIC and CUBIC only probe for bandwidth more aggressively than standard TCP when the loss ratio is small and the capacity is high.

L. Xu, K.Harfoush, I. Rhee, "Binary Increase congestion Control (BIC) for Fast Long-Distance Networks", In Proceedings of IEEE INFOCOM 2004, March 2004.

This performance study indicates that BIC giver good performance on fairness, TCP friendliness, and scalability for high-speed congestion control.

Additional references:

http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/index_files/Page815.htm

3.1.5 TCP Westwood+

Contacts:

Saverio Mascolo
Associate Professor at Politecnico di Bari.
mascolo at poliba.it
http://193.204.59.68/mascolo/

Standardization Status:

None

Principle / Description of Operation:

TCP Westwood+ is a sender side modification of TCP Reno. Westwood+ is actually the original “TCP Westwood” [43] with an enhanced bandwidth estimate. The congestion control algorithm of Westwood is based on bandwidth estimation, which overestimates the available bandwidth in the presence of ACK compression. Westwood+ is also based on the end-to-end bandwidth estimation. It continuously estimates the packet rate of the connection by monitoring the ACK reception rate. When network congestion is experienced, it adaptively sets the congestion window (*cwnd*) and slow start threshold (*ssthresh*).

It additively increases the congestion window (*cwnd*), and when congestion is experienced i.e. when three DUPACKs are received or timeout expires, both the congestion window (*cwnd*) and the slow start threshold (*ssthresh*) are set equal to the estimated bandwidth times the minimum measured round trip time. When a coarse timeout expires, *ssthresh* is again set to the estimated bandwidth times the minimum measured round trip time whereas *cwnd* is set to 1. In this way it avoids the half reduction of *cwnd* of the standard TCP AIMD Algorithm.

Hence this adaptive decrease mechanism makes it more stable. Other than standard TCP, this adaptive approach provides a congestion window that is decreased more in the presence of heavy congestion and less in the presence of light congestion.

Supported operation mode:

Memory to memory (General transport)

Security features:

None

Implementations / API:

ns-2 modules of Westwood+ with the New Reno feature are available at http://193.204.59.68/mascolo/tcp%20westwood/modules.htm

TCP Westwood+ is available in Linux kernel 2.4 and 2.6.

Congestion Control Algorithms:

On ACK reception:

Increase *cwnd* accordingly to the Reno algorithm;

Estimate the available bandwidth (BWE);

When 3 DUPACKs are received:

ssthresh = max(2, (BWE*RTTmin) / seg_size);

cwnd = *ssthresh*;

When coarse timeout expires:

ssthresh = max(2, (BWE*RTTmin) / seg_size);

cwnd = 1;

Fairness:

According to [44], Westwood+ TCP fairly behaves when interacting with Reno TCP and improves the fairness with respect to Reno TCP.

Usable in the public Internet (TCP-Friendly):

By mathematical analysis in [45], it was shown that Westwood+ is friendly towards TCP Reno.

Analytical Model:

When the average packet loss probability p is low, Westwood+ TCP provides the following steady stage throughput according to [44]:

$$T^{West} = \frac{1}{\sqrt{RTT \cdot T_q}} \sqrt{\frac{1-p}{p}}$$

Target Usage Scenario:

The main targets of TCP Westwood+ are High-speed networks. Results indicate that TCP Westwood+ shows good throughput over wireless links and improved fairness in wired networks.

Benchmarking:

S. Mascolo and G. Racanelli, "Testing TCP Westwood+ over Transatlantic links at 10 Gigabit/Second Rate". In proc. PFLDnet 2005, Lyon, February 2005

According to measurements on the DataTag testbed Westwood+ provides significant throughput and fairness improvement over standard TCP. It was also shown that further enhancements can be achieved by modifying the Westwood+ probing phase.

S. Mascolo, L. A. Grieco, R. Ferorelli, P. Camarda, G. Piscitelli, "Performance evaluation of Westwood+ TCP congestion control", Internet performance symposium (IPS 2002), Pages: 93 - 111, January 2004.

In this paper experimental results have been obtained running Linux 2.2.20 implementations of Westwood+, Westwood and Reno TCP over emulated WAN and over Internet connections spanning continental and intercontinental distances. Collected measurements show that the bandwidth

estimation algorithm employed by Westwood+ nicely tracks the available bandwidth. Live Internet measurements also show that Westwood+ TCP improves the goodput. Finally, simulations using ns-2 in controlled scenarios showed similar results.

http://193.204.59.68/mascolo/tcp%20westwood/Results24.htm

Additional References:

http://193.204.59.68/mascolo/tcp%20westwood/tcpwestwood.htm

A. Zanella, G. Procissi, M. Gerla, M.Y. Sanadidi, "TCP Westwood: analytic model and performance evaluation," Proceedings of Globecom 2001, pp.1698-1702, S. Antonio, Texas, December 2001.

3.1.6 Comparison

	HS TCP	Scalable TCP	FAST TCP	BIC & CUBIC	Westwood+
Principle	Modify TCP response function when congestion window is larger than a threshold.	More aggressive in congestion control.	Based on TCP Vegas.	Based on a unique window control function.	Sender side modification of TCP Reno, based on bandwidth estimation.
Operation Mode	Memory-to-memory. Kernel space.	Memory-to-memory. Kernel space.	Memory-to-memory. Kernel space.	Memory-to-memory.	Memory-to-memory.
Authentication	No.	No.	No.	No.	No.
Congestion Control	Yes.	Yes.	Yes.	Yes.	Yes.
Fairness	Need more investigation.	No.	Need more investigation.	Yes.	Yes.
TCP Friendly	No when new TCP response function is triggered.	Need more investigation.	Need more investigation.	Yes.	Yes.
Analytical Model	Yes.	Yes.	Need more investigation.	Yes.	Yes.
Simulation and Implementation	Both	Implementation	Not published.	Both for BIC Simulation for CUBIC	Both

3.2 Protocols based on UDP

3.2.1 UDP and UDP Lite

Contacts:

- L-A. Larzon, Lulea University of Technology;
- M. Degermark, S. Pink, The University of Arizona; L-E. Jonsson, Ericsson;
- G. Fairhurst, University of Aberdeen

Standardization Status:

UDP: RFC 768 [2], status: Standard (STD0006).

UDP-Lite: draft-ietf-tsvwg-udp-lite-02.txt, approved by the IESG for publication as Proposed Standard RFC.

Principle / Description of Operation:

UDP provides a datagram oriented unreliable service; it merely adds the following to the basic IP service: ports to identify individual applications sharing an IP address and a checksum to detect erroneous packets and discard them. Note that UDP is not congestion controlled; irresponsible UDP programming can degrade the performance of TCP senders and make a data stream look like a denial-of-service attack. Ensuring TCP-friendly operation on top of UDP is left up to the application programmer.

Certain applications, e.g. video codecs, can cope with single bit errors – but standard UDP will never pass erroneous data to an application. In UDP Lite, however, having a payload checksum is optional and erroneous data can be passed to applications. This functionality is only useful in combination with link layers that can be tuned not to discard erroneous data, and such link layers are rare. Typically, there is a CRC checksum at the link layer.

Supported operation mode:

Memory to memory (General transport)

Security features:

None

Implementations / API:

UDP is available in every TCP/IP stack; there are no UDP Lite implementations known to the authors. However, it is straightforward to derive UDP Lite from an existing UDP implementation.

Congestion Control Algorithms:

None

Fairness:

None

Usable in the public Internet (TCP-Friendly):

None

Analytical Model:

None

Benchmarking:

None

Target Usage Scenario:

UDP is intended for unreliable datagram oriented transmission - i.e. the block size should be known by the application and a datagram may be lost in transit. UDP Lite extends the UDP functionality with support for erroneous links: it does not necessarily discard the whole packet if a bit error occurs.

3.2.2 Reliable Blast UDP

Contacts:

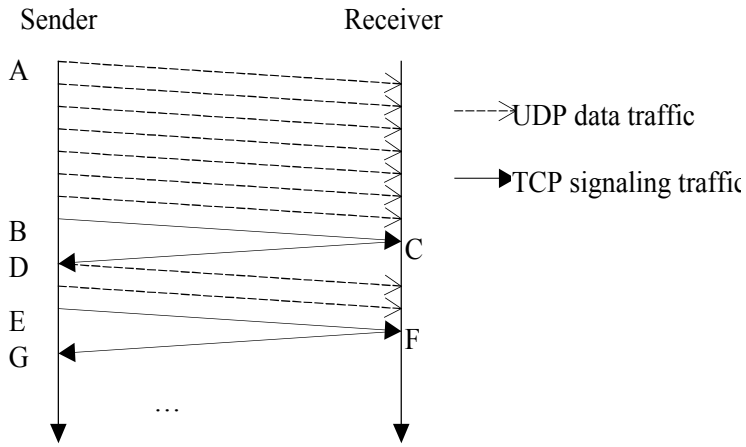
Eric He (eric@evl.uic.edu), Jason Leigh (spiff@evl.uic.edu)

Standardization Status:

None.

Principle / Description of Operation:

Reliable Blast [8,9] 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)

Security features: No

Implementations / API: Provides C++ API

Congestion Control Algorithms:

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.

Usable in the public Internet (TCP-Friendly):

None.

Analytical Model:

The purpose of developing an analytical model for RBUDP is two-fold. Firstly, an equation similar to the "bandwidth × delay product" equation for TCP has been developed, to allow predicting RBUDP performance over a given network. Secondly, it helps systematically identify the factors that influenced the overall performance of RBUDP so that how much benefit any potential enhancement in the RBUDP algorithm might provide can be predicted. A comprehensive predictable performance model for RBUDP is detailed in [8].

Benchmarking:

E. He, Leigh, J., Yu, O., DeFanti T. A., "Reliable Blast UDP : Predictable High Performance Bulk Data Transfer", IEEE Cluster Computing 2002, Chicago, IL 09/01/2002 - 09/01/2002

The results of the implementation of RBUDP show that it performs extremely efficiently over high speed, high bandwidth, dedicated- or Quality-of-Service- enabled networks, such as optically switched networks.

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.

Additional References:

Reference [8]

http://www.evl.uic.edu/cavern/quanta

3.2.3 TSUNAMI

Contacts:

Mark Meiss (mmeiss@indiana.edu)

Standardization status:

None.

Principle / Description of Operation:

The goal of Tsunami is to develop a 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. A single

thread handles both network and disk activity at the sender side whereas the receiver employs separate threads for disk and network activities.

The receiver periodically makes a retransmission request and retransmissions have higher priority than normal sends. The 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. The 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)

Security features:

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.

Implementation / 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.

Usable in the public Internet (TCP-Friendly):

None

Analytical Model:

None

Benchmarking:

Mark R. Meiss "TSUNAMI: A High-Speed Rate-Controlled Protocol for File Transfer"

Results indicate that TSUMANI is really successful on high-speed networks with low-level packet loss in the file transfer domain.

http://www.indiana.edu/~uits/cpo/tsunami/

Target Usage Scenario:

Designed for faster transfer of large files over high-speed networks.

Additional References:

Mark R. Meiss "TSUNAMI: A High-Speed Rate-Controlled Protocol for File Transfer"

http://steinbeck.ucs.indiana.edu/~mmeiss/papers.html

README file of tsunami-2002-12-02 release [10].

http://www.indiana.edu/~anml/anmlresearch.html

3.2.4 UDT - UDP based Data Transfer

Contacts:

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

Standardization status:

Internet Draft: draft-gg-udt-00.txt [34]

Principle / Description of Operation:

UDT is an application level protocol by introducing new reliability and flow/congestion control mechanisms upon UDP. UDT is duplex. Each UDT entity has two parts: the sender and the receiver. The sender sends and retransmits data packets according the control mechanisms. The receiver receives both data and control packets, processes them properly, and may also generate control packets as feedback.

It uses packet based sequencing and ACK sub-sequencing. Since UDT is supposed to be used to transfer bulk data, we assume that most of the data packets can be packed in maximum segment size (MSS). Each UDT data packet is assigned a unique increasing packet sequence number ranging from 0 to 2^31 - 1. Each ACK is assigned a unique ACK sequence number (independent to packet sequence number) ranging from 0 to 2^16 - 1.

The receiver generates an ACK2 immediately after it receives an ACK, with the same ACK sequence number. The peer side can then avoid generating unnecessary ACKs and the ACK/ACK2 pair can be also used to calculate RTT. Additionally, UDT also uses explicit negative acknowledgments (NAK) to report packet loss.

There are four timers used in UDT: 1) ACK timer, 2) NAK timer, 3) rate control timer, and 4) retransmission timer. All the timers are contained in the receiver and are queried after each time bounded UDP receiving.

ACK timer is used to generate periodical selective acknowledgements and the period is constant of 0.01 seconds. NAK timer is used to resend loss reports if the retransmission is not received after an increasing time interval. The NAK period is equal to RTT. Rate control timer is used to generate a periodical rate control event and the period is constant of 0.01 seconds. Retransmission timer is used to trigger data packet retransmission.

Supported operation mode:

Memory to memory (General transport)

Security features:

None.

Implementations / API:

Open source C++ library with socket style API.

Source code: http://sourceforge.net/projects/dataspace API reference is included in the release.

Congestion Control Algorithms:

UDT uses rate-based congestion (rate control) and window based flow control (flow control). Rate control tunes the packet-sending period (STP) every constant interval (rate control timer period, or RCTP), which is 0.01 seconds. Suppose the sender takes t time to pack and send out a packet, then after this packet sending, it sleeps max ((STP - t), 0) time to send next packet.

Once the rate control timer expires, if the packet loss rate during the last RCTP time is less than 0.1, the number of packets to be sent in the next RCTP time (inc) is increased by:

```
if (B <= C)
    inc = 1/MSS
else
    inc = max(10^(ceil(log10((B-C)*MSS*8))) * Beta/MSS, 1/MSS)
```

where B is the estimated link capacity and C is the current sending speed. Both are counted as packets per second. MSS is counted in bytes. Beta is a constant value of 0.0000015.

After inc is calculated, STP can be updated according to:

$$RCTP/STP\_new = RCTP/STP\_old + inc$$

B is estimated using receiver packet pair. Every 16 data packets, the sender sends out the next data packet immediately to form a packet pair. The receiver uses a median filter on the intervals between the two packets in a packet pair to estimate link capacity, and sends it back within ACK.

STP is increased by 1/8 as soon as the sender receives an NAK packet and its largest lost sequence number is greater than the largest sent sequence number when last decrease occurs or the number of NAKs since last rate decrease exceeds an increasing threshold. No data is sent out during next RCTP time after a rate decrease.

UDT also uses a flow control window to limit the number of unacknowledged packets. Once it is time to feed back ACK, the receiver calculates the packet arrival rate (AS) using a median filter on the recent packet arrival intervals it recorded. The flow window size W is updated as:

$$W = ceil\ (W * 0.875 + (RTT + ATP) * AS * 0.125)$$

where ATP is the period of ACK timer, which is 0.01 seconds. The receiver then computes an effective window size as:

$$Effective\_window = max\ (min\ (W, available\ receiving\ buffer), 2).$$

The value of effective window is fed back within ACK to the sender.

The flow window size starts from 2 as slow start, and is updated to the number of acknowledged packets when the sender receives an ACK. Slow start ends when loss occurs or reaches the maximum window size, after which the congestion control algorithm described above starts to work. The packet sending period is 0 during slow start phase and is set to the packet arrival interval when slow start ends.

Fairness:

UDT satisfies max-min fairness on single bottleneck topologies. On multi-bottleneck topologies, any flow is guaranteed to obtain at least half of its fair share according to the max-min rule. Particularly, the fairness of UDT is independent of RTT.

Usable in the public Internet (TCP-Friendly):

When coexisting with concurrent bulk TCP flow, TCP can occupy more bandwidth than UDT does, except in one of the following 3 situations:

- The network BDP is very large that TCP fails to utilize their fair share bandwidth. In this situation, UDT will occupy the bandwidth that TCP cannot utilize.
- The link capacity is so small that UDT's bandwidth estimation techniques do not work optimally. Simulation shows that this threshold link capacity is about 100 kbps.
- In networks where FIFO queues are used along the paths, if the queue size is greater than BDP, TCP's bandwidth share decreases as the queue size increases. However, to reach about equal sharing with UDT, the queue size needed generally exceeds the amount that a practical router/switch will provide.

When short (timewise) web-like TCP flows coexist with a small number of concurrent UDT flows, the effect of UDT on TCP flows is very small.

Analytical Model:

None.

Benchmarking:

Robert L. Grossman, Yunhong Gu, Dave Hanley, Xinwei Hong, Dave Lillethun, Jorge Levera, Joe Mambretti, Marco Mazzucco, and Jeremy Weinberger, Experimental Studies Using Photonic Data Services at IGrid 2002, FGCS, 2003.

Target Usage Scenario:

Designed to be used scenarios where a small number of bulk sources share abundant bandwidth. In other scenarios, e.g., messaging or low BDP networks, UDT can still be used but there may be no improvement in performance.

3.2.5 Comparison

	RBUDP	Tsunami	UDT
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...	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.
Analytical Model	Yes.	No.	No.
Implementation	Available.	Available.	Available.

3.3 TCP-friendly congestion control mechanisms

There is a wealth of research on TCP-friendly congestion control, which did not lead to actual protocol specifications, but rather a set of rules for changing the rate in a way that would be advantageous (e.g., smoother) but still fair towards TCP. In what follows, we give a brief overview of this research; since there was no actual protocol specification, we do not follow the template that was used for the other protocols. Notably, some of these mechanisms may one day be available as an option within the framework of the Datagram Congestion Control Protocol (DCCP) – e.g., there is already a specification for TFRC within DCCP.

Perhaps the simplest commonly known TCP-friendly congestion control mechanism is the “Rate Adaptation Protocol (RAP)”: it imitates the behaviour of TCP in a rate-based manner and, most importantly, does not retransmit lost packets. AIMD behaviour is not achieved by changing a congestion window but by calculation [18]. In [19], RAP is used as part of an end-to-end architecture for real-time multimedia data transmission. While the rate-based nature of RAP may be advantageous for streaming media applications (it does not show the typical stop-and-go behaviour seen with window-based protocols), the important stabilising feature of self-clocking is lost [20].

In the case of TCP and RAP, AIMD corresponds with additively increasing by a factor α = one packet per round-trip-time if no loss occurs and multiplying the rate with a factor β = 0.5 if a packet is lost. The stability of AIMD does not rely on these factors; the old DECbit scheme, for instance, uses a factor of β = 7/8 [33]. Yang and Lam introduce "General AIMD Congestion Control (GAIMD)" [21] as a TCP-friendly generalisation of AIMD; their finding is that it is possible to

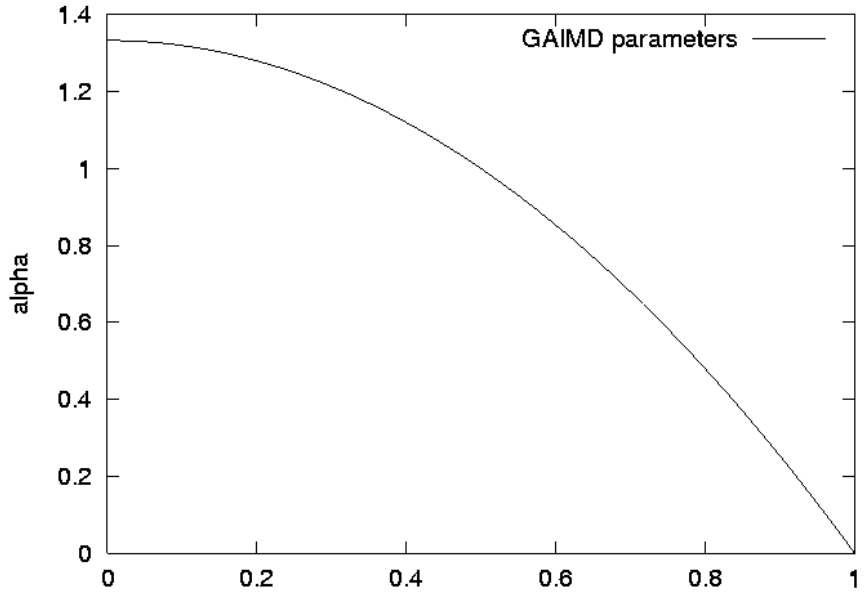


Figure 2: General AIMD parameters (alpha and beta) relation

achieve a TCP-friendly rate with different values for α and β as long as the relation $\alpha = 4(1-\beta^2)/3$ remains valid. This means that in order to remain fair towards TCP, one must consider a trade-off between the responsiveness and the smoothness of an AIMD algorithm – for example, for β = 7/8, α must be set to 0.31. Figure 2 depicts this relationship between the GAIMD increase and decrease factors.

One particular goal of TCP-friendly congestion control is a smoother rate, which is considered an obvious advantage for streaming media applications. One way to achieve this would be to set the GAIMD parameters properly, but there are other methods.

The well known "TCP-friendly Rate Control (TFRC)" protocol uses a more precise version of the equation from [22] which models the steady-state behaviour of TCP:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

Since a TCP-friendly ("TCP-compatible") flow can be defined as a flow that, in steady-state, uses no more bandwidth than a conforming TCP running under comparable conditions [28], such a flow should not exceed the sending rate T, which is a function of the packet size s, round-trip-time R, steady-state loss event rate p and the TCP retransmit timeout value t_{RTO} and is given in bytes/sec.

The most critical of these parameters is the "steady-state loss event rate"; any number of losses within a round-trip-time is considered a single "loss event" in TFRC. The parameter is calculated using the "Average Loss Interval" method, which is a specially weighted moving average of the loss rate. The aggressiveness of TFRC can be tuned by varying the length of the loss interval, but generally, TFRC adapts to traffic fluctuations slowly and achieves a much smoother rate than

RAP. More details can be found in [23]; TFRC is fully specified in [29], extended to multicast in [24] and analyzed in [20].

"TCP Emulation at Receivers (TEAR)" is a sophisticated mechanism where receivers perform calculations that normally, senders would do based on the information in acknowledgements. TEAR assumes "rate independence": the probability of experiencing a packet loss within a certain window of consecutively transmitted packets should not depend on their transmission rate. It achieves a smooth rate because it seldom sends rate updates to the sender; in simulations, the rate of TEAR appeared to be even smoother than the rate of TFRC [30].

There are several other notable TCP-friendly Congestion Control mechanisms. Examples include the "Loss-Delay Based Adjustment Algorithm (LDA+)", which controls the sender rate via RTCP packets, using the Packet Pair approach to increase the rate proportional to the bottleneck bandwidth; it is efficient in terms of network utilisation and fairness towards TCP [25]. Another example is "Binomial Congestion Control" [31], which is a nonlinear generalization of the control laws in [32]. A more complete overview is given in [26], and a qualitative comparison can be found in [27].

3.4 Protocols requiring router support

3.4.1 XCP (eXplicit Congestion control Protocol)

Contacts:

Dina Katabi (dk@mit.edu)

Standardization status:

None.

Principle / Description of Operation:

XCP generalizes the Explicit Congestion Notification (ECN) proposal. Instead of the 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, a 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)

Security features:

None

Implementations/API:

An XCP implementation for 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.

Usable in the public Internet (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.

Analytical 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.

Benchmarking:

Dina Katabi, Mark Handley and Charlie Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", Proceedings on ACM Sigcomm 2002.

XCP maintains good utilization and fairness, has low queuing delay, and drops very few packets in high bandwidth-delay product environments. It was shown to perform very well in a large variety of settings.

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.

Additional References:

http://www.ana.lcs.mit.edu/dina/XCP/

3.4.2 CADPC / PTP

Contacts:

Michael Welzl,

University of Innsbruck.

E-mail: michael.welzl@uibk.ac.at.

Web: http://www.welzl.at

Standardization status:

Internet-draft: draft-welzl-ptp-05.txt (expired) [35]

Principle / Description of Operation:

PTP [16], 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 (and stateless), all calculations are moved 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 well over wireless links. It is slowly responsive in that it utilizes a small amount of feedback, but it shows quick convergence.

Among its 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)

Security features:

None

Implementations / API:

Code is available from http://www.welzl.at/ptp; future releases will be available from this site, too. Currently, there is:

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

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.

Usable in the public Internet (TCP-Friendly):

None.

Analytical 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 (where 1 is with the bottleneck link capacity). With a very small number of users (say, 2 or 3), CADPC is inefficient.

Benchmarking:

CADPC attains high throughput for bulk data transfer while maintaining close to zero loss; since it bases its reaction on occasional explicit signaling messages, traffic dynamics at shorter timescales than the signaling message generation frequency (typically 4 RTTs) are ignored by the mechanism. Thus, it does not work well with very short web-like flows [16].

Target Usage Scenario:

This protocol is intended for 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.

3.5 Others

3.5.1 SCTP (Stream Control Transport Protocol)

Contacts:

The IETF SIGTRAN Working Group: http://www.ietf.org/html.charters/sigtran-charter.html

Standardization status:

Several RFCs, most notably the main specification (RFC 2960 [36], status: Proposed Standard) and an introduction to the protocol (RFC 3286 [37], status: Informational).

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 IESG as a Proposed Standard. 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 [36]) 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. It 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 "head-of-line-blocking" 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.
- 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

Security features:

None

Implementations / API:

There is quite a large number of SCTP implementations available; for instance, a Linux kernel implementation is available from http://sourceforge.net/projects/lksctp

Congestion Control Algorithms:

TCP AIMD

Fairness:

Similar to TCP.

Usable in the public Internet (TCP-Friendly):

Yes.

Analytical Model:

Yes

Benchmarking:

The improvements of SCTP over TCP are manifold, and so are the situations where this protocol could be beneficial. For instance, if an application can cope with out-of-order data, the delay it experiences can be significantly reduced with SCTP in the presence of reordering. Also, the multihoming feature can alleviate the well known problem of slow Internet routing convergence

when a failure occurs. To the best of our knowledge, there is no comprehensive document presenting a thorough analysis of SCTP performance. Note that SCTP generally works just like TCP unless an application makes explicit use of its features.

Target Usage Scenario:

Applicability of SCTP (e.g., in the presence of NAT, and its security implications), is explained at length in RFC 3257 [38]; its service model differs from TCP – simply put, it can be used wherever is TCP is used, but the features above are desirable.

Additional References:

http://www.sctp.org/sctpoverview.html (good for beginners)

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

Referece [39] is interesting to have an overview of multi-homing applications

3.5.2 Datagram Congestion Control Protocol (DCCP)

Contacts:

Eddie Kohler, Mark Handley, Sally Floyd, Jitendra Padhye, ICIR (The ICSI Center for Internet Research) - http://www.icir.org

Standardization status:

DCCP is being developed in the IETF DCCP Working Group:

http://www.ietf.org/html.charters/dccp-charter.html

Several Internet-drafts were published, most notably the main specification draft-ietf-dccp-spec-06.txt [40].

Principle / Description of Operation:

The Datagram Congestion Control Protocol provides a well-defined protocol framework with several options and features (standard ACK format, some security using nonces, Path MTU discovery, identification, negotiation mechanisms, connection set-up and tear-down, ..) for congestion control mechanisms. It does not prescribe a certain congestion control behavior by itself – such specifications are written in separate documents. An additional user guide explains, among other things, how to use of DCCP for real-time media applications. An API to DCCP may, for example, allow an application to specify preferences for negotiable features prior to the initiation of the session – which allows DCCP to perform feature negotiation as per application references without explicit interaction with the application. The idea is to relieve application programmers from the burden of implementing a specially suitable congestion control mechanism and move this logic into the network stack, where it belongs.

Supported operation mode:

Memory to memory (General transport)

Security features:

DCCP has “Identification options”, which provide a way for DCCP endpoints to confirm each others' identities, even after changes of address or long bursts of loss that get the endpoints out of sync. DCCP does not provide cryptographic security guarantees, and attackers that can see every packet are still capable of manipulating DCCP connections inappropriately, but the Identification options make it more difficult for some kinds of attacks to succeed.

Implementations / API:

Links to kernel- and user-level Linux implementations are maintained at
http://www.icir.org/kohler/dcp/

Congestion Control Algorithms:

The main DCCP specification does not prescribe any kind of congestion control; the functionality of such mechanisms is defined in special “profile” documents. Presently, there are two profile definitions, Internet-draft draft-ietf-dccp-ccid2-02.txt for TCP-like congestion control and Internet-draft draft-ietf-dccp-ccid3-02.txt for TFRC congestion control. TFRC is an equation based congestion control mechanism that is TCP-friendly and shows a smoother rate than TCP; more information can be found at http://www.icir.org/tfrc/

Fairness:

The two defined profiles, TCP-like congestion control and TFRC, are TCP-friendly. This defines their fairness. DCCP itself does not define the fairness of mechanisms.

Usable in the public Internet (TCP-Friendly):

The two defined profiles, TCP-like congestion control and TFRC, are TCP-friendly.

Analytical Model:

As DCCP itself merely encapsules congestion control mechanisms, it does not have a performance model.

Benchmarking:

None.

Target Usage Scenario:

DCCP is a general-purpose protocol for any kind of unreliable data traffic, as encountered with streaming media applications or real-time multimedia Internet applications, for example. Its performance depends on the chosen congestion control mechanism.

Additional References:

DCCP maintainer web site: http://www.icir.org/kohler/dcp/

Several Internet-drafts

3.5.3 GridFTP

Contacts:

Bill Allcock (allcock@mcs.anl.gov)

Standardization status:

GGF draft [41] - according to the globus website (http://www.globus.org), it is also intended to submit GridFTP to the IETF.

Principle / Description of Operation:

GridFTP is not a transport protocol by itself. It 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), IETF 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 one 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

Security features:

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

Usable in the public Internet (TCP-Friendly):

Yes

Benchmarking:

GridFTP enables a source to act like a number of TCP flows at the same time, which leads to a somewhat more aggressive form of congestion control than with a single TCP flow. Something similar has been done with a protocol called "MulTCP" [46] – its performance evaluation can therefore be expected to also apply to GridFTP.

Target Usage Scenario:

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

Additional References:

Reference [17]

http://www-fp.globus.org/datagrid/gridftp.html

4. Deployment Considerations

The protocols in this document greatly vary in their degree of deployability and availability. The following is a brief summary from this point of view:

- XCP and CADPC/PTP are perhaps the most critical protocols: they require an update of the software in routers; moreover, all routers along the path (in the case of CADPC/PTP,

at least every other router) must support the protocol. An additional hazard to Internet deployment is the fact that CADPC/PTP is not TCP-friendly and that XCP requires significantly more processing power from routers in order to be TCP-friendly.

- SCTP, DCCP and UDP Lite have been (or are being) developed in the IETF; at least some of them can therefore be expected to be available in some standard TCP/IP stacks in the near future.
- All other TCP-friendly protocols in this document can be deployed in the Internet on an end-to-end basis.
- All other protocols in this document can be deployed within isolated scenarios (on dedicated lines).

Note that HighSpeed TCP changed the notion of "TCP-friendliness" somewhat: while originally, the definition required a protocol not to send more than a conforming TCP would under similar circumstances, HighSpeed TCP only does so when the packet loss ratio is high. Since it only deviates from standard TCP behavior when it cannot cause significant harm (i.e. the link capacity is large and the loss ratio is small), it is still expected to be usable in the public Internet.

In the form that was integrated with HighSpeed TCP, Scalable TCP is similar in that aspect; also, CUBIC follows this principle of falling back to TCP-like behavior in the presence of high loss rates.

5. Security Considerations

The protocols described in this document vary greatly in their degree of security; some have no security support at all, while others (e.g., SCTP) have a significant number of features. Please refer to the specifications of the individual protocols for specifics.

Author Information

Editors:

- Eric He, Electronic Visualization Laboratory, University of Illinois at Chicago. eric@evl.uic.edu
- Pascale Vicat-Blanc Primet, INRIA, France. Pascale.Primet@ens-lyon.fr
- Michael Welzl, University of Innsbruck, Austria. michael.welzl@uibk.ac.at

Contributors:

- 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
- Mathieu Goutelle, ENS Lyon, France. mathieu.goutelle@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

- Muhammad Murtaza Yousaf, University of Innsbruck, Austria. murtaza.yousaf@uibk.ac.at

Intellectual Property Statement

Full Copyright Notice

References

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

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

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

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

[5] Sally Floyd, “HighSpeed TCP for Large Congestion Windows”, RFC 3649, Experimental, December 2003.

[6] S. Iren and P. Amer, “The transport layer: tutorial and survey”, ACM Computing survey, vol. 31, No. 4, December 1999.

[7] Volker Sander et al., “Networking Issues of Grid Infrastructures", GGF Grid High Performance Networking Research Group, draft, September 2003.

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

[9] 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.

[10] README file of tsunami-2002-12-02 release.

http://www.indiana.edu/~anml/anmlresearch.html

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

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

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

[14] Cheng Jin, David X. Wei and Steven H. Low, “FAST TCP: motivation, architecture, algorithms, performance”, Proceedings of IEEE Infocom, March 2004.

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

[16] Michael Welzl, "Scalable Performance Signalling and Congestion Avoidance", Kluwer Academic Publishers, 2003. ISBN: 1-4020-7570-7

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

[18] Reza Rejaie, Mark Handley, and Deborah Estrin, “RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet“, Proceedings of IEEE Infocom 1999, New York City, New York, 21.-25. 3. 1999.

[19] Reza Rejaie, Mark Handley, and Deborah Estrin, “Quality Adaptation for Congestion Controlled Video Playback over the Internet“, Proceedings of ACM SIGCOMM 1999, Cambridge, MA., September 1999.

[20] Deepak Bansal, Hari Balakrishnan, Sally Floyd, Scott Shenker, “Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms“, Proceedings of ACM SIGCOMM 2001, San Diego, CA, August 2001.

[21] Y. Richard Yang and Simon S. Lam, “General AIMD Congestion Control“,Technical Report TR-2000-09, Department of Computer Sciences, The University of Texas at Austin, May 9, 2000.

[22] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation“,Proceedings of ACM SIGCOMM 1998, Vancouver, British Columbia, September 1998.

[23] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer, “Equation-Based Congestion Control for Unicast Applications“,Proceedings of ACM SIGCOMM 2000.

[24] Jörg Widmer and Mark Handley, “Extending Equation-based Congestion Control to Multicast Applications“,Proceedings of ACM SIGCOMM 2001, San Diego, CA, August 2001.

[25] Dorgham Sisalem and Henning Schulzrinne, “The Loss-Delay Based Adjustment Algorithm: A TCP-friendly Adaptation Scheme“,Proceedings of NOSSDAV 1998.

[26] Jörg Widmer, Robert Denda, and Martin Mauve, “A Survey on TCP-Friendly Congestion Control“,IEEE Network Magazine, Special Issue “Control of Best Effort Traffic“ Vol. 15, No. 3, May 2001.

[27] Yang Richard Yang, Min Sik Kim, and Simon S. Lam, “Transient Behaviors of TCP-friendly Congestion Control Protocols“,Proceedings of IEEE Infocom 2001, Anchorage, Alaska, April 2001.

[28] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, “Recommendations on Queue Management and Congestion Avoidance in the Internet”, RFC 2309, April 1998.

[29] M. Handley, S. Floyd, J. Padhye, J. Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification”, RFC 3448, January 2003.

[30] Injong Rhee, Volkan Ozdemir, and Yung Yi, “TEAR: TCP emulation at receivers – flow control for multimedia streaming”, Technical Report, Department of Computer Science, NCSU. Available from: http://www.csc.ncsu.edu/faculty/rhee/export/tear_page/

[31] Deepak Bansal and Hari Balakrishnan, “Binomial Congestion Control Algorithms”, Proceedings of INFOCOM 2001, Anchorage, Alaska, April 2001.

[32] D. Chiu and R. Jain, “Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks”, Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14.

[33] Raj Jain and K. K. Ramakrishnan, “Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals and Methodology”, Proceedings of Computer Networking Symposium, Washington, D. C., April 11-13 1988, pp. 134-143.

[34] Y. Gu and R. Grossman, “UDT: A Transport Protocol for Data Intensive Applications”, Internet-draft draft-gg-udt-00.txt (work in progress), 7. Jan. 2004.

[35] Michael Welzl, "The Performance Transparency Protocol (PTP)", Internet-draft draft-welzl-ptp-05.txt (work in progress), June 2001. Expired.

[36] R. Stewart, Q. Xie et al., "Stream Control Transmission Protocol", RFC 2960, October 2000. Status: Proposed Standard.

[37] L. Ong, J. Yoakum, "An Introduction to the Stream Control Transmission Protocol (SCTP)", RFC 3286, May 2002. Status: Informational.

[38] L. Coene, "Stream Control Transmission Protocol Applicability Statement", RFC 3257, April 2002. Status: Informational.

[39] 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.

[40] Eddie Kohler, Mark Handley, and Sally Floyd, “Datagram Congestion Control Protocol (DCCP)”, Internet-draft draft-ietf-dccp-spec-06.txt (work in progress), February 2004.

[41] W. Allcock, J. Bester, J. Bresnahan, S. Meder, S. Tuecke, “GridFTP: Protocol Extensions to FTP for the Grid”, Global Grid Forum Draft

[42] Injong Rhee, and Lisong Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", Third International Workshop on Protocols for Fast Long-Distance Networks, Feb 3,4 2005

[43] R. Wang, M. Valla, M.Y. Sanadidi and M. Gerla, “Using Adaptive Bandwidth Estimation to provide enhanced and robust transport over heterogeneous networks”, 10th IEEE International Conference on Network Protocols (ICNP 2002), Paris, France, Nov. 2002.

[44] L. A. Grieco, S. Mascolo, "Performance Comparison of Reno, Vegas and Westwood+ TCP Congestion Control", ACM SIGCOMM Computer Communication Review, April 2004.

[45] Grieco L. A., Mascolo S., "Westwood TCP and easy RED to improve Fairness in High Speed Networks", IFIP/IEEE Seventh International Workshop on Protocols For High-Speed Networks", PfHSN02, April 2002.

[46] Jon Crowcroft and Philippe Oechslin: "Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP", ACM SIGCOMM Comput. Commun. Rev., Volume 28, Number 3, 1998.