A Study of Protocols for Grid Computing Environment

Suresh Jaganathan *whosuresh@gmail.com*
Associate Professor
Department of Computer Science and Engineering
Sri Sivasubramania Nadar College of Engineering
Chennai, Tamilnadu, India

Srinivasan A *asrini30@gmail.com*
Professor
Department of Computer Science and Engineering
Misrimal Navajee Munoth Jain Engineering College
Chennai, Tamilnadu, India

Damodaram A *damodarama@jntuh.ac.in*
Professor
Department of Computer Science and Engineering
Jawaharlal Nehru Technological University
Hyderabad, Andra Pradesh, India

Abstract

Using grid systems efficiently has to face many challenges. One of them is the efficient exchange of data between distant components exacerbated by the diversity of existing protocols for communicating participants. Grid computing will reach its vast potential if and only if, the underlying networking infrastructure is able to transfer data across quite long distances in a very effective manner. Experiences show that advanced distributed applications executed in existing large scale computational grids are often able to use only a small fraction of available bandwidth. The reason for such a poor performance is the TCP, which works only in low bandwidth and low delay networks. Several new transport protocols have been introduced, but a very few are widely used in grid computing applications, these protocols can be categorized in three broad categories viz. TCP based, UDP based and Application layer protocols. We study these protocols and present its performance and research activities that can be done in these protocols.

Keywords: Communication Protocol, Grid Computing, Bandwidth Delay Networks and Performance

1. INTRODUCTION

GRID computing [1, 2] is a technology for coordinating large scale resource sharing and problem solving among various autonomous group. Grid technologies are currently distinct from other major technical trends such as internet, enterprise distributed networks and peer to peer computing. Also it has some embracing issues in QoS, data management, scheduling, resource allocation, accounting and performance.

Grids are built by various user communities to offer a good infrastructure which helps the members to solve their specific problems which are called a grand challenge problem. A grid consists of different types of resources owned by different and typically independent organizations which results in heterogeneity of resources and policies. Because of this, grid based services and applications experience a different resource behavior than expected. Similarly, a distributed infrastructure with ambitious service put more impact on the capabilities of the interconnecting networks than other environments.

Grid High Performance Network Group [3] works on network research, grid infrastructure and development. In their document the authors listed six main functional requirements, which are

considered as mandatory requirements for grid applications. They are: i) high performance transport protocol for bulk data transfer, ii) performance controllability, iii) dynamic network resource allocation and reservation, iv) security, v) high availability and vi) multicast to efficiently distribute data to group of resources.

New trends of data communication protocols are needed for grid, because of these reasons, i) networks emerging with long fat network [LFN], high bandwidth and long delay such as LambdaGrid [4], OptiPuter [5] , CANARIE [6] and sensor networks for grid [7], ii) communication patterns which are shifting from one-to-one communication to many-to-one and many-to-many communication and iii) there is an unique type of communication needs in transport characteristics for each grid application, such as rate, loss ratio and delay. For all these data communications, standard transport protocols such as TCP [8] and UDP [9] are not always sufficient, for example, traditional TCP reacts adversely when there is an increase in bandwidth and delay leading to poor performance in high BDP networks [10]. It is still a challenge for grid applications to use the bandwidth available, due to the limitations of current and today's network transport protocols.

For instance, a remote digital image processing application may use unreliable fixed rate data transfer, where as a fast message parsing application can demand reliable data transfer with minimum delay, and a wireless grid application can demand for minimized data traffic to prolong battery life. As TCP acts as reliable transport protocol and UDP with no guarantees, these two protocols cannot provide the optimal mix of communications attributes for different grid application. More number of transport protocols have been developed and proposed over the last 3 decades. In this paper, we study the various variants of such transport protocols based on TCP and UDP, and compare the protocols in various points for grid computing. Each protocols is reviewed based on the i)operation, ii)operation mode, iii)implementation, iv)congestion control, v)fairness, vi)throughput, vii)TCP friendly, viii)security, xi)quality of service and x)usage scenario.

Section II highlights the issues in designing high performance protocols and briefs how TCP and UDP play an important role in implementing grid computing protocols. Section III surveys the various protocols, for bulk data transfer, high speed, and high bandwidth delay and with high performance with TCP as base. Section IV surveys UDP based protocols. Section V deals with application layer protocols and Section VI concludes paper with summary.

2. ISSUES IN COMMUNICATION PROTOCOLS FOR GRID COMPUTING

The emerging high-performance grid can have a wide range of network infrastructures and different communication patterns for different types of applications, these combinations and factors made researchers to develop a new data communication protocols especially for grid environments. For these applications, available standard protocols (TCP and UDP) are not sufficient, because of their properties and lack of flexibility. Performance of TCP not only depends on transfer rate, but also on the product of round-trip delay and transfer rate. This *bandwidth*delay* product measures the amount of data that would fill the pipe, and that amount of data is the buffer space required at sender and receiver side to obtain maximum throughput on the TCP connection over the path. TCP performance problems arise when the *bandwidth*delay* product becomes large. Three fundamental performance problems with the TCP over high bandwidth delay network paths are; i) window size limit, ii) recovery from losses, and iii) round-trip measurement.

In grid high performance research document [3], it summarizes the networking issues available in grid applications and gives some consideration for designing a protocol for grid computing, they are: i) slow start, ii) congestion control, iii) ACK clocking, iv) connection setup and teardown and in [11] author lists some parameters which relates to TCP performance in high speed networks, they are: i) cwnd increase function, ii) responsiveness requirements, iii) scalability and iv) network dynamics. In [12] some general set of transport attributes are listed for grid computing, i) connection oriented vs. connectionless, ii) reliability, iii) latency and jitter, iv) sequenced vs. unsequenced, v) rate shaping vs. best effort, vi) fairness and vii) congestion control. The rapid

growth of network infrastructures and communication methods made the grid community to demand enhanced and diverse transport functionality. Because of these growth in network environment, QoS guarantees and its properties depends on these parameters, i) type of link (dedicated or shared), ii) communication methods (point-to-point or multipoint-to-point) and iii) application requirements.

2.1. Need of High Performance Communication Protocol

2.2.1. Increasing Capabilities

Increasing in computing power has a doubling time of 18 months, storage device capacity doubles every 12 months and communication speed doubles every 9 months. The difference in rate of increase creates a situation in all areas, although significant power available, because of traditional implementation tradeoff method changes. This increasing capability has resulted in work to define new ways to interconnect and manage computing resources.

2.2.2. New Application Requirements

New applications are being developed in physics, biology, astronomy, visualization, digital image processing, meteorology etc. Many of the anticipated applications have communication requirements between a small numbers of sites. This presents a very different requirement than presented by “typical” Internet use, where the communication is among many sites. Applications can be defined in three classes: 1) lightweight “classical” Internet applications (mail, browsing), 2) medium applications (business, streaming, VPN) and 3) heavyweight applications (e-science, computing, data grids, and virtual presence). The total bandwidth estimate for all users of each class of network application is 20 Gb/sec for the lightweight Internet, 40 Gb/sec for all users of the intermediate class of applications and 100 Gb/sec for the heavyweight applications. Note that the heavyweight applications use significantly more bandwidth than the total bandwidth of all applications on the classical Internet. Different application types value different capabilities. Lightweight applications give importance to interconnectivity, middleweight applications to throughput and QoS, while the heavyweight applications to throughput and performance.

An example of heavyweight application, the Large Hadron Collider (LHC) in CERN, has requirements well beyond what is available now. The LHC produce and distribute gigantic amounts of data. The data rates from the LHC in CERN are estimated to be in the order 100 to 1500 Megabytes per second. Dedicated fiber will carry data from the Atlas Collector in CERN to the initial storage and computing devices around the world, bypassing the Internet. Another example of heavyweight application are the CAVE Research Network, a work being done at the Electronic Visualization Lab (EVL) at the University of Illinois at Chicago which uses very large communication rates to implement total immersion projects. Other examples include the Electron Microscopy work being done at the University of California San Diego (UCSD) and Osaka.

TCP works well on the commodity internet, but has been found to be inefficient and unfair to concurrent flows as bandwidth and delay increase [13, 14, 15, 16]. Its congestion control algorithm needs very long time to probe the bandwidth and recover from loss in high BDP links. Moreover, the existence of random loss on the physical link, the lack of a buffer on routers, and the existence of concurrent bursting flows prevent TCP from utilizing high bandwidth with a single flow. Furthermore, it exhibits a fairness problem for concurrent flows with different round trip times (RTTs) called RTT bias. The success of TCP is mainly due to its stability and the wide presence of short lived, web-like flows on the Internet. However, the usage of network resources in high performance data intensive applications is quite different from that of traditional internet applications. First, the data transfer often lasts a very long time at very high speeds. Second, the computation, memory replication, and disk I/O at the end hosts can cause bursting packet loss or time-outs in data transfer. Third, distributed applications need cooperation among multiple data connections. Finally, in high performance networks, the abundant optical bandwidth is usually shared by a small number of bulk sources. These constraints made the researchers to design a new transport protocol for high performance domains.

2.2. Congestion Control

Moving bulk data quickly over high-speed data network is a requirement for many applications. These applications require high bandwidth link between network nodes. To maintain the stability of internet, all applications should be subjected to congestion control. TCP is well-developed, extensively used and widely available Internet transport protocol. TCP is fast, efficient and responsive to network congestion conditions, but one objection to using TCP congestion control is that TCP's AIMD congestion back-off algorithm [17], which is too abrupt in decreasing the window size, thus it hurts the data rate.

The performance of the congestion control system, TCP algorithm and the link congestion signal algorithm has many facets and these variables can impact the Quality of Service (QoS). A variety of merits described they are:

Fairness: The Jain index [18] is a popular fairness metric that measures how equally the sources sharing a single bottleneck link. A value of 1 indicates perfectly equal sharing and smaller values indicate worse fairness.

Throughput: Throughput is simply the data rate, typically in Mbps, delivered to the application. For a single source this should be close to the capacity of the link. When the BDP is high, that is, when the link capacity or RTT or both are high, some protocols are unable to achieve good throughput.

Stability: The stability metric measures the variations of the source rate and/or the queue length in the router around the mean values when everything else in the network is held fixed. Stability is typically measured as the standard deviation of the rate around the mean rate, so that a lower value indicates better performance. If a protocol is unstable, the rate can oscillate between exceeding the link capacity, and thus resulting in poor delay jitter and throughput performance.

Responsiveness: measures how fast a protocol reacts to a change in network operating conditions. If the source rates take long time to converge to a new level, say after the capacity of the link changes, either the link may become underutilized or the buffer may overflow. The responsiveness metric measures the time or the number of round trips to obtain the right rate.

Queuing delay: Once congestion window is greater than the BDP, the link is well utilized and however, if the congestion window is increased more, queuing delay builds up. Different TCP and AQM protocol combinations operate on how to minimize the Queuing Delay.

Loss recovery: packet loss can be a result because of overflowing buffers, which indicates network congestion, and also of transmission error, such as bit errors over a wireless channel. It is desirable that, when packet loss occurs due to transmission error, the source continues to transmit uninterrupted. However, when the loss is due to congestion, the source should slow down. Loss recovery is typically measured as the throughput that can be sustained under the condition of a certain random packet loss caused by transmission error. Loss based protocols typically cannot distinguish between congestion and transmission error losses.

In the past few years, more number of TCP variant were developed, that address the under-utilization problem most notably due to the slow growth of TCP congestion window, which makes TCP unfavorable for high BDP networks. Table 1 list some TCP variants addressing the conservative approach of TCP to update its congestion window under congestion condition.

Congestion Control Approaches	TCP Variants
Loss-based TCP congestion control	High Speed TCP, BIC TCP, Scalable TCP, CUBIC TCP and Hamilton TCP
Delay-based congestion control	TCP-Vegas, Fast-TCP
Mixed loss-delay based TCP congestion control	Compound TCP ,TCP Africa
Explicit congestion Notification	XCP

TABLE 1: TCP Variants based on various congestion control approaches

Loss based high speed algorithms are aggressive to satisfy bandwidth requirement but this aggressiveness causes TCP unfairness and RTT unfairness. Delay based approaches provide

RTT fairness but it is difficult to meet TCP fairness, where in third approach, a synergy of delay based and loss based approach address the problem in the two approaches.

2.3. Previous Work

Delivering communication performance in high bandwidth-delay product network is a major research challenge. Compared to shared, packet-switched IP networks, the key distinguishing characteristics of grid computing communications are: i) no internal network congestion and very small end system congestion, ii) small numbers of endpoints and iii) very high speed links i.e. more than 10GB.. Traditional TCP and its variants were developed for shared networks, where the bandwidth on internal links is a critical and has limited resource. As such, available congestion control techniques manage internal network contention, providing a reasonable balance of non-aggressive competition and end-to-end performance. This results in slow start, causing TCP to take a long time to reach full bandwidth when RTT is large and takes a long time to recover from packet loss because of its AIMD control law. In [19, 20], a simulation-based performance analysis of HighSpeed TCP is presented and the fairness to regular TCP is analyzed. In [21], the author deals with the performance of Scalable TCP and analyzes the aggregate throughput of the standard TCP and Scalable TCP based on an experimental comparison. In [22], the performance of different TCP versions, such as HighSpeed TCP, Scalable TCP, and FAST TCP, are compared in an experimental test-bed environment. In all cases, the performance among connections of the same protocol sharing bottleneck link is analyzed and different metrics such as throughput, fairness, responsiveness, stability are presented.

To provide good transport performance for network with high bandwidth-delay product optical network links, researchers have proposed many delay based TCP variations and recently, rate-based reliable transport protocols are proposed based on UDP. They explicitly measure packet loss, and adjust transmission rates in response. A number of delay based TCP variants are proposed for shared, packet-switched networks and few user level rate-based protocols, targeting different network scenarios and communication methods. Evaluation of these protocols are done using point to point single flow, parallel point to point flows, and multipoint convergent flows. From the evaluation results, rate-based protocols deliver high performance in high bandwidth-delay product networks, but because of their aggressiveness, there occurs high rate of packet loss when multiple flows exist.

3. TCP BASED PROTOCOLS FOR GRID NETWORKS

New challenges for TCP have been addressed by several research groups in the last 3 decades and, as a result, a number of new TCP variants have been developed. An overview and study of protocols that are designed for high speed bulk data transfer in high bandwidth delay networks are given here, and the TCP variants analyzed in this paper are summarized in Figure 1.

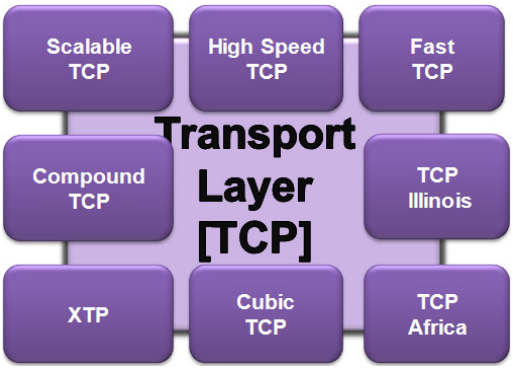


FIGURE 1: TCP Variants

Emerging networks bring new challenges. First, the new architecture, heterogeneous networks, mobile and wireless environments exhibit different network characteristics requiring more attention on the dynamical aspects of the operation. As a consequence of this, the dynamic behavior of TCP flows has to be taken into consideration. On the one hand, it is obvious that the dynamic effects have significant impact on the performance and throughput of the TCP flows [23]. On the other hand, the fairness also needs to be reconsidered from the aspects of dynamic behavior. The performance analyses of various TCP variants are included in many papers [24, 25, 26].

3.1. Scalable TCP (S-TCP)

Performance of today's network is to be increased due to applications like distributed simulation, remote laboratories, and multi gigabyte data transfers. TCP fails to capture the available bandwidth in high performance network, because of two reasons: i) size of socket buffer at the end-hosts which limits the transfer rate and the maximum throughput and ii) packet loss causing multiplicative cwnd value decrease and additive increase of cwnd when in absence of loss, reduces the average throughput. To address these issues researchers focused in three approaches: i) modifying available TCP, ii) parallel TCP and iii) automatic buffer sizing.

Modifying TCP congestion control scheme and also in routers, can lead to significant benefits for both applications and networks. Parallel TCP connection increases the aggregate throughput, but fails in maintaining fairness and also in increasing the window rate. Several researchers have proposed TCP modifications, mainly on congestion control schemes making TCP more effective in high performance paths. Kelly proposed Scalable TCP [21]. The main feature in the Scalable TCP is it adopts constant window increase and decrease factors in case of congestion and multiplicative increase in absence of congestion.

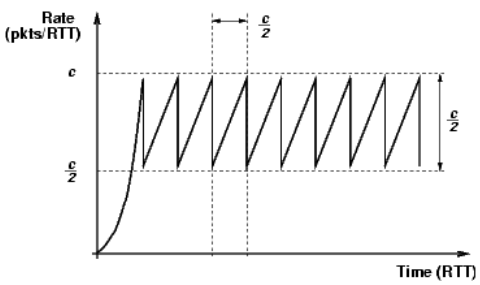
Scalable TCP is designed in such a way that it automatically switches to traditional TCP stacks in low bandwidth and to incremental method at the time of high bandwidth available. STCP is a simple sender side modification to TCP congestion control, and it employs Multiplicative Increase Multiplicative Decrease (MIMD) technique. Using Scalable TCP, better utilization of a network link with the high bandwidth-delay product can be achieved. If STCP is mixed with regular TCP then STCP dominates the bandwidth for sufficiently large bandwidth-delay product region, which results in unfriendliness towards standard TCP. Congestion Window [cwnd] value is changed according to the equation shown below:

$$cwnd = cwnd + 0.01 \qquad for\ each\ ack\ received\ and\ no\ loss$$

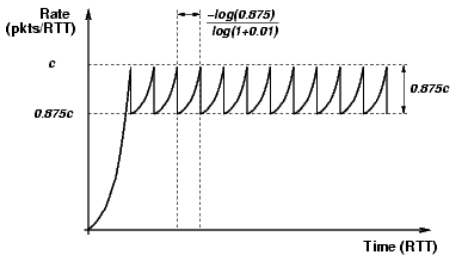
$$cwnd = 0.875 * cwnd \qquad on\ each\ loss\ event$$

Scalable TCP response function is calculated as $W_{scalable} = \frac{0.0745}{p}$ where 'p' is packet loss rate.

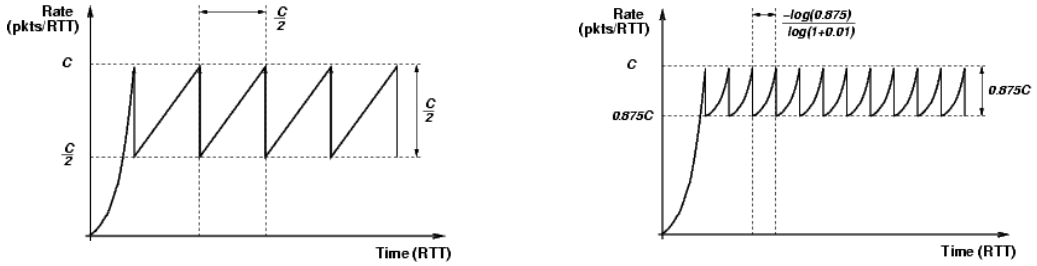
Figure 2 shows the probing results of traditional and Scalable TCP. The main difference between the two depends on round trip time and rate, traditional TCP are proportional to sending rate and round trip time, where are in Scalable TCP it is proportional only to round trip time and not depends on rate.



a. Traditional TCP with small capacity



b. Scalable TCP with small capacity



c. Traditional TCP with large capacity **d.** Scalable TCP with large capacity

FIGURE 2: Scaling properties Traditional TCP vs. Scalable TCP

Table 2 shows the recovery time for standard TCP congestion control and Scalable TCP congestion control for various sending rates and packet loss in less than one RTT.

Rate	Recovery Time	
	Standard TCP	**Scalable TCP**
1Mbps	1.7s	2.7s
10Mbps	17s	3.0s
100Mbps	2mins	4.7s
1Gbps	28mins	2.7mins
10Gbps	4hrs 43mins	1 hrs 24mins

TABLE 2: Recovery Times

The recovery time after packet loss is 13.42RTT, i.e. proportional to the RTT and independent of congestion window size. An STCP connection can recover even a large congestion window in a short time and so that it makes efficient use of the bandwidth in high-speed networks. For e.g. for a TCP connection with 1500 byte MTU and RTT value of 200ms, then for 10Gbps network, congestion window recovery time after packet loss for STCP is 1hrs 24mins whereas that for Standard TCP is approximately 4hrs 43mins. STCP can be used in high performance environment because of i) its efficient congestion control, ii) improved performance in high BDP and iii) increased throughput, but it does has fairness issue.

3.2. High Speed TCP (HS-TCP)

Networking applications such as multimedia web streaming seek more bandwidth and bulk-data transfer. These applications often operate over a network with high BDP, so the performance over these networks is a critical issue [27]. Recent experiences indicate that TCP has difficulty in utilizing high-speed connections, because of this, network applications are not able to take utilize the available bandwidth in high speed network [28]. The amount of packets that are needed to fill a gigabit pipe using present TCP is beyond the limit of currently achievable in fiber optics and the congestion control is not so dynamic, also most users are unlikely to achieve even 5 Mbps on a single stream wide-area TCP transfer, even the underlying network infrastructure can support rates of 100 Mbps or more.

HSTCP (HighSpeed TCP) is a variant to the TCP, which is specifically designed for use in high speed, high bandwidth network. Congestion management allows the protocol to react and to recover from congestion and operate in a state of high throughput yet sharing the link fairly with other traffic.

The performance of a TCP connection is dependent on the network bandwidth, round trip time, and packet loss rate. At present, TCP implementations can only reach the large congestion window, necessary to fill a pipe with a high bandwidth delay product, when there is an exceedingly low packet loss rate. Otherwise, random losses lead to significant throughput deterioration when the product of the loss probability and the square of the bandwidth delay are

larger than one. The HighSpeed TCP for large congestion window [29] was introduced as a modified version of TCP congestion control mechanism. It is designed to have different responses in very low congestion event rate, and also to have the standard TCP responses with low packet loss rates. Further, TCP behavior is unchanged when there is mild to heavy congestion and doesn't increase the congestion collapse.

HighSpeed TCP's modified response function comes in effect with higher congestion window, and TCP behavior is unchanged when there is heavy congestion and doesn't introduce any congestion collapse. HSTCP uses three parameters, W_L, W_H, and P_H. To ensure TCP compatibility, when the congestion window is W_L at most, HSTCP uses the standard TCP response function, and uses the HSTCP response function when congestion window is greater than W_L. When the average congestion window is greater than W_L, then the response function is calculated as follows:

$$W = (\frac{p^s}{p_L^s})W_L, \; where \; s = \frac{(logW_H - logW_L)}{(logP_H - logP_L)}$$

HSTCP keeps average congestion window W_H and W_L, when packet loss rates are P_H and P_L, respectively. Recommended parameters are: $W_L = 38$, $W_H = 83000$ and $P_H = 10^{-7}$. Even though there is a loss rate in high-speed environments, it still allows acceptable fairness for the HighSpeed response function when compared with Standard TCP environments with packet drop rates of 10^{-4} or 10^{-5}. The HSTCP response function is computed as shown below:

$$W_{highspeed} = \frac{0.12}{p^{0.835}}$$

HSTCP is more aggressive than standard TCP and a HighSpeed TCP connection would receive ten times the bandwidth of a standard TCP in an environment with packet drop rate of 10^{-6}, which is unfair. The HighSpeed TCP response function is represented by new additive increase and multiplicative decrease parameters and these parameters modify the function parameters according to congestion window value.

HighSpeed TCP performs well in high-speed long-distance links. It falls back to standard TCP behavior if the loss ratio is high. In case of burst traffic, its link utilization is improved but at the same time there are some issues regarding fairness.

3.3. TCP Africa

The limitations of standard TCP are apparent in networks with large bandwidth-delay-products, which are becoming increasingly common in the modern Internet. For example, supercomputer grids, high energy physics, and large biological simulations require efficient use of very high bandwidth Internet links, often with the need to transfer data between different continents. The desired congestion window used by standard TCP is roughly equal to the BDP of the connection. For high bandwidth-delay-product links, this desired congestion window is quite high, as high as 80,000 packets for a 10 Gbps link with 100 ms RTT.

TCP's deficiencies with a purely loss based protocol leads to super-aggressive protocols, which raises fairness and safety concerns. By contrast, delay based protocols, which make use of the wealth of in-formation about the network state that is provided by packet delays, can indeed achieve excellent steady state performance, as well as minimal self-induced packet losses. These delay-responsive protocols do not cause enough packet drops to the non-delay-responsive protocols to force them to maintain only their fair share. TCP-Africa, an Adaptive and Fair Rapid Increase Congestion Avoidance mechanism for TCP is proposed [30], to solve this problem which is a new delay sensitive congestion avoidance mode, which has scalability, aggressive behavior.

TCP-Africa, a new delay sensitive two-mode congestion avoidance rule for TCP, promises excellent utilization, efficiency, and acquisition of available bandwidth, with significantly improved safety, fairness, and RTT bias properties. This new protocol uses an aggressive, scalable window

increase rule to allow quick utilization of available bandwidth, but uses packet round trip time measurements to predict eminent congestion events. Once the link is sensed to be highly utilized, the protocol reacts by resorting to the conservative congestion avoidance technique of standard TCP.

TCP-Africa is a hybrid protocol that uses a delay metric to determine whether the bottleneck link is congested or not. In the absence of congestion, the *fast mode,* where it uses aggressive, scalable congestion avoidance rule. In the presence of congestion, the *slow mode*, it switches to the more conservative standard TCP congestion avoidance rule.

TCP-Africa, in its scalable "fast" mode, quickly grabs available bandwidth. Once the delay increases, the protocol senses that the amount of available bandwidth is getting small. So, it switches to a conservative "slow" mode.

TCP-Africa detects congestion using the following metric:

$$\frac{W(aRTT - minRTT)}{aRTT} \geq \propto$$

The quantity $(aRTT - minRTT)$ gives us an estimate of the queuing delay of the network. Since the overall round trip time is $minRTT + (aRTT - minRTT)$, the quantity $(aRTT - minRTT)/aRTT)$ is the proportion of the round trip time that is due to queuing delay rather than propagation delay. TCP maintains an average sending rate of $\frac{W}{aRTT}$ packets per second and in case of TCP algorithm it is $\frac{W(aRTT-minRTT)}{aRTT}$ an estimate for the number of packets that the protocol currently has in the queue. The $\propto$ parameter is a constant, usually set as a real number larger than one. The choice of α determines how sensitive the protocol is to delay. By tracking this metric, `TCP Africa can detect when its packet are beginning to en-queue at the bottleneck link, and thus, determine an opportune time to switch into slow growth mode.

The congestion avoidance steps followed by the protocol are:

if $(W(aRTT - minRTT) < \propto x\ aRTT)$

$\{W = W + fast\_increase(W)/W\}$

else

$\{W = W + 1/W$

The function $fast\_increase(W)$ is specified by a set of modified increase rules for TCP. The Table 3 shows the experimental results of TCP Africa.

RTT Ratio	Throughput Ratio	Packet Losses
1	1.0132	440
2	2.3552	581
3	3.6016	529
6	8.4616	336

TABLE 3: TCP Africa Experimental Results

Analysis of this protocol shows, switching to a one packet per RTT rate can greatly decrease the frequency of self inflicted packet loss. Thus, TCP-Africa can be quick to utilize available bandwidth, but slow when it comes to inducing the next congestion event.

3.4. Fast TCP

Advances in computing, communication, and storage technologies in global grid systems started to provide the required capacities and an effective environment for computing and science [31]. The key challenge to overcome the problem in TCP is, using Fast TCP congestion control algorithm which does not scale to this advancement. The currently deployed TCP implementation is a loss-based approach. It uses additive increase multiplicative decrease (AIMD) and it works

well at low speed, but additive increase (AI) is too slow and multiplicative decrease (MD) too drastic leading to low utilization of network resources. Moreover, it perpetually pushes the queue to overflow and also discriminates against flows with large RTTs. To address these problems, Fast TCP was designed, which adopts delay based approach.

Fast TCP has three key differences: i) it is an equation based algorithm [32], ii) for measuring congestion, it uses queuing delay as a primary value, iii) has stable flow dynamics and achieves weighted fairness in equilibrium that it does not penalize long flows. The advantages of using queuing delay as the congestion measure are, i) queuing delay is estimated more accurately than loss probability, and also loss samples provide coarser information than queuing delay samples. This makes easier stabilize a network into a steady state with a high throughput and high utilization and ii) the dynamics of queuing delay have the right scaling with respect to network capacity.

In Fast TCP, congestion is avoided by using maximum link utilization, which can be attained by adjusting the source's sending rate so that resource is shared by all TCP connections. Fast TCP uses two control mechanisms for achieving its objective, they are: i) dynamically adjusting the send rate and ii) using aggregate flow rate to calculate congestion measure. Under normal network conditions, Fast TCP periodically updates the congestion window 'w' based on the average RTT according to below equation

$$w = min\left\{2w, (1-\gamma)w + \gamma\left(\frac{baseRTT}{RTT}w + \alpha\right)\right\}$$

where $\gamma \in (0,1)$, RTT is the current average round-trip time, baseRTT is the minimum RTT, and α is a protocol parameter that controls fairness and the number of packets each flow buffered in the network [33]. It is proved that, in the absence of delay, this algorithm is globally stable and converges exponentially to the unique equilibrium point where every bottleneck link is fully utilized and the rate allocation is proportionally fair. Table 4 shows the experimental results of FAST TCP for implementation in Linux v2.4.18.

Flow	Transfer [Gb]	Utilization	Delay [ms]	Throughput [Mbps]
1	380	95%	180	925
2	750	92%	180	1797
7	15300	90%	85	6123
9	3750	90%	85	7940
10	21650	88%	85	8609

TABLE 4: FAST TCP experimental results

Fast TCP performs well in these areas: i) throughput, ii) fairness, iii) robust and iv) stability, 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.

3.5. TCP-Illinois

Several new protocols have been introduced to replace standard TCP in high speed networks. For all the protocols, the increase in window size initially slow, when the network is in absence of congestion, window size is small and in case congestion, the window size becomes large. As a result, the window size curve between two consecutive loss events is convex. This convex nature is not desirable. First, the slow increment in window size, at the time of absence of congestion is inefficient. Second, the fast increment in window size, at the time of congestion causes heavy loss. Heavy congestion causes more frequent timeouts, more synchronized window bakeoffs, and is more unfair to large RTT users. So the main problem with AIMD algorithm is the convexity of the window size curve. An ideal window curve should be concave, which is more efficient and avoids heavy congestion. To overcome and rectify this problem, a general AIMD algorithm is modified in such a way that it results in a concave window curve.

TCP-Illinois a variant of TCP congestion control protocol [34], developed at the University of Illinois at Urbana-Champaign, targeted at high-speed, long distance and long fat networks. It achieves high average throughput than the standard TCP, by modifying congestion control in sender side. It allocates the network resource fairly, a TCP friendly protocol and provides incentive for TCP users to switch.

TCP-Illinois is a loss-delay based algorithm, which uses packet loss as the primary congestion signal, and it uses queuing delay as the secondary congestion signal to adjust the pace of window size [35]. Similar to the standard TCP, TCP-Illinois increases the window size W by α / W for each acknowledgment, and decreases W by βW for each loss event. Unlike the standard TCP, α and β are not constants. Instead, they are the functions of average queuing delay d_a $\alpha = f_1(d_a)$, $\beta = f_2(d_a)$. In detail

$$\alpha = f_1(d_a) = \begin{cases} \alpha_{max} & if\ d_a \leq d_1 \\ \dfrac{k_1}{k_2 + d_a} & otherwise \end{cases}$$

$$\beta = f_2(d_a) = \begin{cases} \beta_{min} & if\ d_a \leq d_2 \\ k_3 + k_4 d_a & if\ d_2 < d_a < d_3 \\ \beta_{max} & otherwise \end{cases}$$

$where\ \alpha_{max} = \frac{k_1}{k_2 + d_1}$, $\beta_{min} = k_3 + k_4 d_2$, $\beta_{max} = k_3 + k_4 d_3$.

Suppose d_m is the maximum average queuing delay and denoted as $\alpha_{min} = f_1(d_m) = \frac{k_1}{k_2 + d_m}$, where $k_1 = \frac{(d_m - d_1)\,\alpha_{min}\,\alpha_{max}}{\alpha_{max} - \alpha_{min}}$, $k_2 = \frac{(d_m - d_1)\,\alpha_{min}}{\alpha_{max} - \alpha_{min}} - d_1$, $k_3 = \frac{(\beta_{min} d_3 - \beta_{max} d_2)}{d_3 - d_2}$, $k_4 = \frac{\beta_{max} - \beta_{min}}{d_3 - d_2}$.

TCP-Illinois increases the throughput much more quickly than TCP when congestion is far and increases the throughput very slowly when congestion is imminent. As a result, the average throughput achieved is much larger than the standard TCP. It also has many other desirable features, like fairness, compatibility with the standard TCP, providing incentive for TCP users to switch, robust against inaccurate delay measurement.

3.6. Compound TCP

The physicists at CERN LHC conduct physics experiments that generate gigabytes of data per second, which are required to be shared among other scientists around the world. These bulk data has to move quickly over high speed data network, currently most of the applications use the TCP for this purpose. TCP is a reliable data transmission with congestion control algorithm, which takes care of congestion collapses in the network by adjusting the sending rate according to the available bandwidth of the network but in the to-day's Internet environment, TCP substantially underutilizes network bandwidth over high-speed and long distance networks.

Compound TCP (CTCP) is a synergy of delay-based and loss-based approach [36]. The sending rate of Compound TCP is controlled by both sender and receiver components. This delay-based component rapidly increases sending rate at the time of underutilized, but retreats slowly in a busy network when bottleneck queue is built.

Compound TCP integrates a scalable delay-based component into the standard TCP congestion avoidance algorithm [37, 38]. This scalable delay-based component has a fast window increase function when the network is underutilized and reduces the sending rate when a congestion event is sensed. Compound TCP maintains the following state variables, $cwnd(congestion\ window)$, $dwnd\ (delay\ window)$ and $awnd\ (receiver\ advertised\ window)$. TCP sending window is calculated as $W_{tn} = \min\ (cwnd + dwnd, awnd)$ where $cwnd$ is updated in the same way as controlled by standard TCP, whereas in CTCP, on arrival of an ACK, $cwnd$ is modified as:

$$cwnd = cwnd + \left(\frac{1}{cwnd + dwnd}\right)$$

CTCP efficiently uses the network resource and achieves high bandwidth utilization. In theory, CTCP can be very fast to obtain free network bandwidth, by adopting a rapid increase rule in the delay-based component, e.g. multiplicative increase.

CTCP has similar or even improved RTT fairness com-pared to regular TCP. This is due to the delay-based component employed in the CTCP congestion avoidance algorithm. When compared to other TCP variants, fairness value is good for CTCP. Because of the delay-based component, CTCP can slowly reduce the sending rate when the link is fully utilized. Hence, there is no self-induced packet loss when compared to standard TCP, maintained good fairness to other competing TCP flows and opted for high performance computing protocol.

3.7. CUBIC TCP

Yet another variant of TCP, called CUBIC that enhances the fairness property, scalability and stability [39]. The main feature of CUBIC is that its window growth function is defined in real time so that its growth will be independent of RTT.

CUBIC TCP is an enhanced version of Binary Increase Congestion Control shortly BIC [40]. It simplifies the BIC window control function and is TCP-friendliness and RTT fairness. As the name of the protocol represents, the window growth function of CUBIC is a cubic function in terms of the elapsed time accounting from the last loss event. The protocol keeps the window growth rate independent of RTT, which keeps the protocol TCP friendly under short and long RTTs. The congestion period of CUBIC is determined by the packet loss rate alone. As TCP's throughput is defined by the packet loss rate as well as RTT, the throughput of CUBIC is defined only by the packet loss rate. Thus, when the loss rate is high and/or RTT is short, CUBIC can operate in a TCP mode. More specifically, 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 from 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 constant multiplication decrease factor applied for window reduction at the time of loss event (i.e., the window reduces to βW_{max} at the time of the last reduction).

An important feature of CUBIC is that it keeps the epoch fairly long without losing scalability and network utilization. Generally, in AIMD, a longer congestion epoch means slower increase (or a smaller additive factor). However, this would reduce the scalability of the protocol, and also the network would be underutilized for a long time until the window becomes fully open.

Unlike AIMD, CUBIC increases the window to W_{max} very quickly and then holds the window there for a long time. This keeps the scalability of the protocol high, while keeping the epoch long and utilization high. This feature is unique in CUBIC.

3.8. Xpress Transport Protocol (XTP)

In near future, data transfer over network increases, thus requiring high speed protocols, standard TCP fails to utilize the available bandwidth, due to its congestion control algorithm. Some variants of congestion control algorithm are proposed, which improves the throughput, but has poor fairness. Xpress Transport Protocol (XTP) is a transport layer protocol for high-speed networks developed by XTP Forum [41] to replace TCP. XTP provides protocol options for error control, flow control, and rate control. Instead of separate protocols for each type of communication, XTP controls packet exchange prototype to produce different models, e.g. reliable datagram's, transactions, unreliable streams, and reliable multicast connections.

XTP provides for the reliable transmission of data in an inter-networked environment, with real time processing of the XTP protocol i.e., the processing time for incoming or outgoing packets is

no greater than transmission time. XTP contains error, flow and rate control mechanisms similar to those found in other more modem transport layer protocols in addition to multicast capability. Timer management is minimized in XTP as there is only one timer at the receiver, used in closing the context. Address translation, context creation, flow control, error control, rate control and host system interfacing can all execute in parallel. The XTP protocol is considered a lightweight protocol for several reasons. First, it is a fairly simple yet flexible algorithm. Second, packet headers are of fixed size and contain sufficient information to screen and steer the packet through the network. The core of the protocol is essentially contained in four fixed-sized fields in the header — KEY, ROUTE, SEQ and the command word. Additional mode bits and flags are kept to a minimum to simplify packet processing.

Xpress Transport Protocol (XTP) is a next generation protocol, designed to be used in high-speed networks and for multimedia applications. It meets the data transfer requirements of many real time systems and has flexible transfer capabilities, in conjunction with various underlying high performance network technology. It provides good support for event-type distributed systems, in both LAN and other internetwork topologies. XTP has been attracting, as a most suitable protocol for high speed network applications. As it provides a wide range of options to the applications to select the different services they need, which makes XTP very attractive particularly for multimedia and other applications, when constrained by the limitations of current protocols.

XTP is able to increase the performance due to its efficient control and error handling algorithms. When the network is more congested or in high speed networks, XTP shows significant higher throughput, maintaining the bandwidth under control and minimizing the CPU utilization. And also this protocol is designed so as to provide more flexibility to distributed applications, supports priority, transmission control rate, selective retransmission, works under UDP with reliable datagram, has rate and burst control, error and flow control, selective acknowledgement, which makes another suitable protocol for high performance computing.

3.9. Summary

Loss-based congestion control algorithms, like HS-TCP and Scalable TCP use packet loss as primary congestion signal, increase window size for each ACK and decrease window size for packet loss. The advantage of delay-based algorithms is that they achieve better average throughput, since they can keep the system around full utilization. As a comparison, the loss based algorithms purposely generate packet losses and oscillate between full utilization and under utilization. However, existing delay-based algorithms suffer from some inherent weaknesses. First, they are not compatible with standard TCP. FAST TCP yields non-unique equilibrium point if competing with other TCP variants. TCP Illinois gives the option to switch over standard TCP, but it increases the throughput slowly. CTCP has a very good embedded congestion control and avoidance algorithm in standard TCP, takes care of fairness value, has increased throughput, but fails in high speed high bandwidth networks and particularly for multimedia applications. XTP comes in picture, where it supports and delivers good transfer rate for multimedia applications in high speed networks. Table 5 presents the summary of TCP based protocols for high performance grid computing environment.

Protocols	Contributors	Year	Changes Done	Remarks
S-TCP	Tom Kelly	2003	Sender Side congestion Control	Exploration is needed
HS-TCP	E. Souza, D. Agarwal	2004	Congestion Control	Loss ratio is high
TCP Africa	Ryan King, Richard Baraniuk, Rudolf Riedi	2005	Delay based CC	Simulation Results using ns2.27 + More overhead than TCP Reno
Fast TCP	Cheng Jin et.al.	2006	Delay based CC	No realistic experiments

TCP Illinois	S. Liu, T. Basar, and R. Srikant	2007	C-AIMD Algorithm	Only Simulated Results using NS2 & Heterogenous users
CTCP	Tan, K. Song, J. Zhang, Q. Sridharan, M.	2007	synergy of delay-based and loss-based Congestion Control	Implemented in windows OS+ effectively utilizes the link capacity
CUBIC TCP	Sangtae Ha, Injong Rhee, Lisong Xu	2008	Modifies the window value using a cubic function	Linux Kernel 2.6.23.9
XTP	Diogo R. Viegas and Rodrigo Mario A. R. Dantas and Michael A. Bauer	2009	Implements multicast options in standard protocol	Multicast Communication Protocol

TABLE 5: Summary of TCP Based Protocols for HPC

4. UDP BASED PROTOCOLS FOR GRID NETWORKS

UDP-based protocols provide much better portability and are easy to install. Although implementation of user level protocols needs less time to test and debug than in kernel implementations, it is difficult to make them as efficient, because user level implementations cannot modify the kernel code, there may be additional context switches and memory copies. At high transfer speeds, these operations are very sensitive to CPU utilization and protocol performance. In fact, one of the purposes of the standard UDP protocol is to allow new transport protocols to be built on top of it. For example, the RTP protocol is built on top of UDP and supports streaming multimedia. In this section we study some UDP based transport protocol for data intensive grid applications. Figure 3 gives some of the UDP variants that are analyzed in this paper.

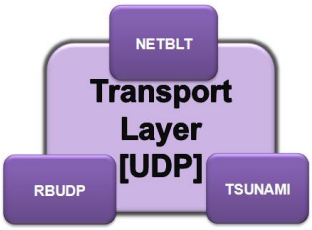


FIGURE 3: UDP Variants

4.1 NETBLT

Bulk data transmission is needed for more applications in various fields and it is must for grid applications. The major performance concern of a bulk data transfer protocols is high throughput. In reality, achievable end-to-end throughput over high bandwidth channels is often an order of magnitude lower than the provided bandwidth. This is because it is often limited by the transport protocols mechanism, so it is especially difficult to achieve high throughput and reliable data transmission across long delay, unreliable network paths.

Throughput is often limited by the transport protocol and its flow control mechanism and it is difficult to achieve high throughput, reliable data transmissions across long delay network paths. To design a protocol, with high throughput, robust in long delay networks, a new flow control mechanism has to be implemented. NETBLT checks the validity of the rate based flow control mechanism, and gains implementation and testing experience with rate control.

Generally speaking, errors and variable delays are two barriers to high performance for all transport protocols. A new transport protocol, NETBLT [42, 43], was designed for high throughput, bulk data transmission application and to conquer the two barriers. NETBLT was first proposed in 1985 (RFC 969). The primary goal of the NETBLT is to get high throughout and robust in a long delay and high loss of network. Seeing the problems with window flow control

and timers, NETBLT builders decided that the goal can be achievable only by employing a new flow control mechanism by implementing and testing experience with rate control.

NETBLT works by opening a connection between two clients (the sender and the receiver) transferring data in a series of large numbered blocks (buffers), and then closing the connection. NETBLT transfer works as follows: the sending client provides a buffer of data for the NETBLT layer to transfer. NETBLT breaks the buffer up into packets and sends the packets using the internet datagram's. The receiving NETBLT layer loads these packets into a matching buffer provided by the receiving client. When the last packet in that buffer has arrived, the receiving NETBLT part will check to see if all packets in buffer have been correctly received or if some packets are missing. If there are any missing packets, the receiver requests to resend the packets. When the buffer has been completely transmitted, the receiving client is notified by its NETBLT layer. The receiving client disposes the buffer and provides a new buffer to receive more data. The receiving NETBLT notifies the sender that the new buffer is created for receiving and this continues until all the data has been sent. The experimental results show that NETBLT can provide the performance predictability than in parallel TCP.

NETBLT protocol is tested in three different networks (RFC1030). The first network is a token ring with 10Mbit/second, served as a reference environment for possible performance. The second network is Ethernet with better performance and third in high bandwidth network. NETBLT's performance in Ethernet allowed the researchers to account it for high bandwidth network. The test used several parameters such as: i) burst interval, ii) burst size, iii) buffer size, iv)data packet size and v) number of outstanding buffers. Table 6 gives the performance details of NETBLT protocols, in three different networks.

Networks →	Token Ring	Ethernet	Wideband
Transfer Size	2 – 5 Million Bytes	2 – 5 Million Bytes	500 – 750 Kilobytes
Data Packet Size	1990 Bytes	1438 Bytes	1432 Bytes
Buffer Size	19900 Bytes	14380 Bytes	14320 Bytes
Burst Size	5 Packets	5 Packets	5 Packets
Burst Interval	40 ms	30 ms	55ms

TABLE 6: Performance test results for NETBLT

In NETBLT, one important area needs to be explored, i.e. dynamic selection and control of NETBLT's transmission parameters (burst size, burst interval) has to be studied. Some research work on dynamic rate control is going on, but more work needs to be done before integrating NETBLT in high performance computing.

4.2 Reliable Blast UDP (RBUDP)

RBUDP [44] is designed for extremely high bandwidth, dedicated or quality of service enabled networks, which require high speed bulk data transfer which is an important part. RBUDP has two goals: i) keeping the network buffer full during data transfer and ii) avoiding TCP's per packet interaction and sending acknowledgements at the end of a transmission.

Figure 4 shows the data transfer scheme of RBUDP. In the data transfer phase (1 to 2 in figure 4 on sender side) RBUDP sends the entire payload to the receiver, using the receiver specified rate. Since the full payload is sent using UDP which is an unreliable protocol, some datagram's may be lost, therefore the receiver has to keep a tally of datagram's received in order to determine which has to be retransmitted. At the end, the sender sends an end signal by sending '*DONE*' via TCP (3 in figure 4 on receiver side) to the receiver and receiver acknowledges by sending a total number of received datagram's sequence numbers to the sender (4 in figure 4 on sender side). The sender checks the acknowledgement and resends the missing datagram's to the receiver.

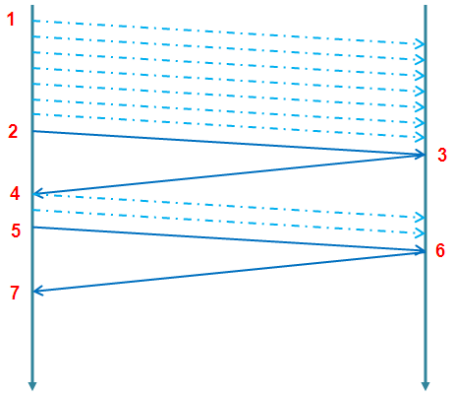


FIGURE 4: Time Sequence Diagram of RBUDP

In RBUDP, the most important input parameter is the sending rate of UDP blasts. To minimize loss, the sending rate should not be larger than the bandwidth of the bottleneck link. Tools such as Iperf [45] and Netperf [46] are typically used to measure the bottleneck bandwidth. There are 3 version of RBUDP available:

i) Version 1: without scatter/gather optimization - this is naive implementation of RBUDP where each incoming packet is examined and then moved.

ii) Version 2: with scatter/gather optimization this implementation takes advantage of the fact that most incoming packets are likely to arrive in order, and if transmission rates are below the maximum throughput of the network, packets are unlikely to be lost.

iii) Version 3: Fake RBUDP this implementation is the same as the scheme without the scatter/gather optimization except that the incoming data is never moved to application memory.

The implementation result of RBUDP shows that it performs very efficiently over high speed, high bandwidth, and Quality of Service enabled networks such as optically switched network. Also through mathematical modeling and experiments, RBUDP has proved that it effectively utilizes available bandwidth for reliable data transfer.

4.3 TSUNAMI

A reliable transfer protocol, Tsunami [47], is designed for transferring large files fast over high speed networks. Tsunami is a protocol which uses inter-packet delay for adjusting the rate control instead of sliding window mechanism. UDP is used for sending data and TCP for sending control data. The goal of Tsunami is to increase the speed of file transfer in high speed networks that use standard TCP. The size of the datagram is avoided in the initial setup and length of file is exchanged in the initial step [connection setup]. A thread is created at each end [sender and receiver], which handles both network and disk activity. The receiver sends a retransmission signal which has higher priority, at the time of datagram loss and at last it sends an end signal. In Tsunami, the main advantage is, the user can configure the parameters such as size of datagram, threshold error rate, size of transmission queue and acknowledgement interval time. The initial implementation of the Tsunami protocol consists of two user-space applications, a client and a server. The structure of these applications is illustrated in Figure 5.

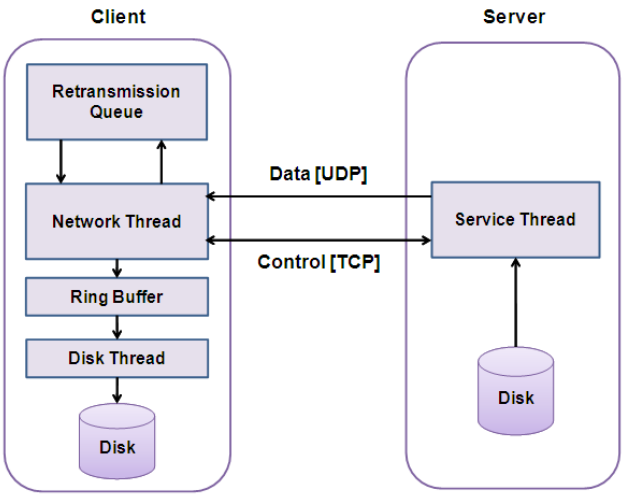


FIGURE 5: Tsunami Architecture

During a file transfer, the client has two running threads. The network thread handles all network communication, maintains the retransmission queue, and places blocks that are ready for disk storage into a ring buffer. The disk thread simply moves blocks from the ring buffer to the destination file on disk. The server creates a single thread in response to each client connection that handles all disk and network activity. The client initiates a Tsunami session by connecting it to the TCP port of the server. Upon connection, the server sends a random data to the client. The client checks the random data by using XOR with a shared secret key and calculates a MD5 check sum, then transmits it to the server. The server does the same operation and checks the check sum and if both are same, the connection is up. After performing the authentication and connection steps, the client sends the name of file to the server. In the server side, it checks whether the file is available and if it is available, it sends a message to client. After receiving a positive message from server, client sends its block size, transfer rate, error threshold value. The server responds with the receiver parameters and sends a time-stamp. After receiving the timestamp, client creates a port for receiving file from the server and server sends the file to the receiver.

Tsunami is an improvement of Reliable Blast UDP in two points. First, Tsunami receiver makes a retransmission request periodically (every 50 packets) and it doesn't wait until finishing of all data transfer, then it calculates current error rate and sends it to the sender. Second, Tsunami uses rate based congestion control. Tsunami has best performance in networks with limited distance, when it comes to long distance network, bandwidth utilization goes down, absence of flow control affects its performance and issues like fairness and TCP friendliness has to be studied.

4.4 Summary

Three UDP-based transfer protocols have much better performance than TCP. Among them, Tsunami has the best performance and RBUDP also performs well when file size is small in size. However, these protocols lack in some ways. First the bandwidth utilization is comparatively low (<70%) which makes not suitable for high speed long distance networks. Second, both RBUDP and Tsunami doesn't estimate bandwidth and uses some third party tools such as Iperf and Netperf. RBUDP has no congestion control and flow control. Tsunami has no flow control, which makes more packet loss in receiver side because of not knowing the receiver's receiving capacity. These existing UDP-based protocols are still to be enhanced in many ways, first, protocol with high throughput for bulk data is to be designed. Then, the issues like fairness and TCP-friendliness are to be considered. Table 7 presents the summary of UDP based protocols for high performance grid computing environment.

Protocols	Contributors	Year	Changes Done	Remarks
NETBLT	David D Clark, Mark L Lambert, Lixia Zhang	1987	New flow control mechanism	High Throughput
RBUDP	Eric He, Jason Leigh, Oliver Yu, Thomas A. DeFanti	2002	SACK + pipe full	Depends on Iperf & Netperf
TSUNAMI	Mark R. Meiss	2002	Threads	User Level ease, no flow control

TABLE 7: Summary of UDP Based Protocols for HPC

5. APPLICATION LAYER BASED PROTOCOLS FOR GRID NETWORKS

The need for high performance data transfer services is becoming more and more critical in today's distributed data intensive computing applications, such as remote data analysis and distributed data mining. Although efficiency is one of the common design objectives in most network transport protocols, efficiency often decreases as the bandwidth delay product (BDP) increases. Other considerations, like fairness and stability, make it more difficult to realize the goal of optimum efficiency. Another factor is that many of today's popular protocols were designed when bandwidth was only counted in bytes per second, so performance was not thoroughly examined in high BDP environments. Implementation also becomes critical to the performance as the network BDP increases. A regular HTTP session sends several messages per second, and it does not matter when message processing is delayed for a short time. Whereas, in data intensive applications, the packet arrival speed can be as high as 100 packets per seconds and any such delay matters. The protocol needs to process each event in a limited amount of time and inefficient implementations can lead to packet loss or time-outs. People in the high performance computing field have been looking for application level solutions. One of the common solutions is to use parallel TCP connections [48] and tune the TCP parameters, such as window size and number of flows. However, parallel TCP is inflexible because it needs to be tuned on each particular network scenario. Moreover, parallel TCP does not address fairness issues. In this section we review some application level protocols for high performance computing purposes especially for grid applications. Figure 6 gives some of the applications layer protocols analyzed in this paper.

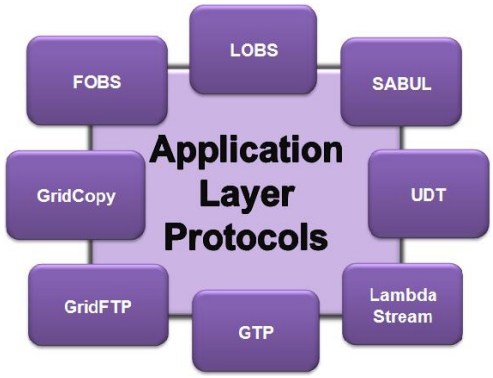


FIGURE 6: Application Layer Variants

5.1. Fast Object Based File Transfer System (FOBS)

A FOBS is an efficient, application level data transfer system for computer grids [49]. TCP is able to detect and respond to network congestion and because of its aggressive congestion control mechanism results in poor bandwidth utilization even when the network is lightly loaded.

FOBS is a simple, user level communication mechanism designed for large scale data transfers in the high bandwidth, high delay network environment typical of computational Grids. It uses

UDP as the data transport protocol, and provides reliability through an application level acknowledgement and retransmission mechanism.

FOBS employs a simple acknowledgement and retransmission mechanism where the file to be transferred is divided into data units called chunks. Data is read from the disk, transferred to the receiver and writes in the disk in units of chunks. Each chunk is subdivided into segments, and the segments are further subdivided into packets. Packets are 1470 bytes (within the MTU of most transmission medium), and a segment consists of 1000 packets. The receiver maintains a bitmap for each segment in the current chunk depicting received / not received status of each packet in the segment. These bitmaps are sent from the data receiver to the data sender at internals dictated by the protocols and triggers (at a time determined by the congestion / control flow algorithm) a retransmission of the lost packets. The bitmaps are sent over a TCP sockets.

FOBS is a lightweight application level protocol for grids. It is UDP based and acts as a reliable transfer protocol. FOBS uses rate controlled congestion control mechanism, which reduces packet loss and outperforms TCP. For the further usage in high performance computing, congestion control mechanism can be still enhanced to give optimal performance.

5.2. Light Object Based File Transfer System (LOBS)

LOBS is a mechanism for transferring file in high performance computing networks which are optimized for high bandwidth delay network, especially for computational grids. Heart of this environment is the ability of transferring vast data in an efficient manner. LOBS rectify the problem found in the TCP for the data transfer mechanism for grid based computations. In LOBS the increase in performance optimization is done and the order of data delivered is not considered. A LOBS is built directly on top of FOBS [50]. The TCP window size plays a vital role for achieving best performance in high bandwidth delay networks; this leads to tune the size at runtime. In LOBS, the size of TCP window is tuned using different approach, i.e. using UDP stream. UDP is used because of these reasons: i) user level not kernel level, ii) to avoid multiplexing TCP streams in kernel level and iii) to provide user level enhancements.

The basic working concept of LOBS is, it creates threads in sender part for controlling its data buffer, to read file from the disk and fills the data buffer. Once the buffer is full, it is transferred to the client over the network. When the data is in transfer mode, the other threads start reading the data from the file and fill the data buffer and the steps are repeated again until the full file is transferred. In LOBS, the goal is to make use of network I/O operation and the disk I/O operation to the largest extent as possible.

Two protocols closely related to LOBS are RBUDP [44] and SABUL [51, 52]. Primary differences between these two protocols are how loss of packet is interpreted and how to minimize the packet loss impacts affecting the behavior of the protocol. SABUL assumes that packet losses indicate congestion, and it reduces the rate based on the perceived congestion, where as in LOBS it is assumed that some packet loss is inevitable and does not make any changes in the sender rate. Primary difference between LOBS and RBUDP is based on the type of network for which the protocols is designed.

LOBS is used to transfer large files between two computational resources in a grid and this mechanism is lightweight and has lesser functionalities than GridFTP [59]. LOBS supports the primary functionality required for computation grids (i.e. fast and robust file transfer).

5.3. Simple Available Bandwidth Utilization Library (SABUL)

SABUL [51, 52] is an application level library, designed for data intensive grid application over high performance networks and to transport data reliably. SABUL uses UDP for the data channel and it detects, retransmits dropped packets. Using TCP as a control channel reduces the complexity of reliability mechanism. To use available bandwidth efficiently, SABUL estimates the bandwidth available and recovers from congestion events as soon as possible. To improve performance, SABUL does not acknowledge every packet, but instead acknowledges packets at

constant time interval which is called as selective acknowledgement [53]. SABUL is designed to be fair, so that grid applications can employ parallelism and also designed in a way so that all flows ultimately reach the same rate, independent of their initial sending rates and of the network delay. SABUL is designed as an application layer library so that it can be easily deployed without any changes in operating systems network stacks or to the network infrastructure.

SABUL is a reliable transfer protocol with loss detection and retransmission mechanism. It is light in weight with small packet size and less computation overhead, hence can be deployed easily in public networks and is also TCP friendly. In SABUL, both the sender and the receiver maintain a list of the lost sequence numbers sorted in ascending order. The sender always checks the loss list first when it is time to send a packet. If it is not empty, the first packet in the list is resent and removed; otherwise the sender checks the number of unacknowledged packets flow size, and if not, it packs a new packet and sends it out. The sender then waits for the next sending time decided by the rate control. The flow window serves the job of limiting the number of packet loss upon congestion when TCP control reports about the occurrence of delay and the maximum window size is set as $Bandwidth * RTT\ (use\ SYN\ instead\ of\ RTT\ if\ SYN > RTT)$.

After each constant synchronization (SYN) time, the sender triggers a rate control event that updates the inter packet time. The receiver receives and reorders data packets. The sequence numbers of lost packets are recorded in the loss list and removed when the resent packets are received. The receiver sends back ACK periodically if there is any newly received packet. The ACK interval is the same as SYN time. The higher the throughput the less ACK packets generated. NAK is sent once loss is detected. The loss will be reported again if the retransmission has not been received after k*RTT, where k is set to 2 and is incremented by 1 each time the loss is reported. Loss information carried in NAK is compressed, considering that loss is often continuous. In the worst case, there is 1 ACK for every received DATA packet if the packet arrival interval is not less than the SYN time, there are M/2 NAKs when every other DATA packet gets the loss for every M sent DATA packets.

5.4. UDP based Data Transfer Protocol (UDT)

UDT [54, 55] is a high performance data transfer and is an alternative data transfer protocol for the TCP when its performance goes down. Goal of UDP is to overcome TCP's inefficiency in BDP networks and in connection oriented unicast and duplex networks. The congestion control module is an open source, so that different control algorithms can be deployed apart from native or default control algorithm based on AIMD.

UDT is more complex than Reliable Blast UDP and Tsunami, but similar to TCP. UDT which is based on UDP adds reliability, congestion control and flow control mechanism. For reliability, UDT uses selective acknowledgement (SACK) at a fixed interval and sends negative acknowledgement (NACK) once a loss is detected. In case of congestion control, UDT adopts DAIMD (AIMD with decreasing increase) algorithm to adjust sending rate. To estimate the link capacity it uses receiver based packet pairs and flow control to limit the number of unacknowledged packets.

Rate control tunes the inter-packet time at every constant interval, which is called SYN. The value of SYN is 0.01 seconds, an empirical value reflecting a trade off among efficiency, fairness and stability. For every SYN time, the packet loss rate is compared with the last SYN time, when it is less than a threshold then the maximum possible link Bit Error Rate (BER) and the number of packets that will be sent in the next SYN time is increased by the following equation,

$$inc = \max\left(10^{\lceil \log_{10}((B-C)*MTU*8)\rceil} * \frac{\beta}{MTU}, \frac{1}{MTU}\right)$$

where B is the estimated bandwidth and C is the current sending rate, both in number of packets per second, β is a constant value of 0.0000015. MTU is the maximum transmission unit in bytes, which is the same as the UDT packet size. The inter-packet time is then recalculated using the total estimated number of sent packets during the next SYN time. The estimated bandwidth B is

probed by sampling UDT data packet pairs and is designed for scenarios where numbers of sources share more bandwidth than expected. In other scenarios, e.g., messaging or low BDP networks, UDT can still be used but there may be no improvement in performance.

UDT adopts complex congestion control algorithm and flow control algorithm which makes not suitable for bulk data transfer in dedicated networks and fairness issue, TCP friendliness also has to be studied.

5.5. Lambda Stream

Network researchers have reached a consensus that the current TCP implementations are not suitable for long distance high performance data transfer. Either TCP needs to be modified radically or new transport protocols should be introduced. Long Fat Networks (LFNs), where the round-trip latencies are extremely high, this latency results in gross bandwidth under-utilization when TCP is used for data delivery. Several solutions have been proposed, one such solution is to provide revised versions of TCP with better utilization of the link capacity. Another solution is to develop UDP-based protocols to improve bandwidth usage. The Simple Available Bandwidth Utilization Library (SABUL [51], Tsunami [47], Reliable Blast UDP (RBUDP) [44] and the Group Transport Protocol (GTP) [58] are few recent examples.

LambdaStream is an application-layer transport protocol [56]. Correspondingly, key characteristics of LambdaStream include a combination loss recovery and a special rate control, which avoids packet loss inherent in other congestion control schemes. To efficiently utilize bandwidth and quickly converge to a new state, the protocol sets the initial sending rate as the quotient of the link capacity over the maximum number of flows, which is easily obtained in a dedicated network.

It adapts the sending rate to dynamic network conditions while maintaining a constant sending rate whenever possible. One advantage of this scheme is that the protocol avoids deliberately provoking packet loss when probing for available bandwidth, a common strategy used by other congestion control schemes. Another advantage is that it significantly decreases fluctuations in the sending rate. As a result, streaming applications experience small jitter and react smoothly to congestion. Another important feature is that the protocol extends congestion control to encompass an end-to-end scope. It differentiates packet loss and updates the sending rate accordingly, thus increasing throughput.

LambdaStream builds on experiences from a high performance networking protocol called Reliable Blast User Datagram Protocol (RBUDP) which transports data using UDP and TCP for control packets. In LambdaStream, the congestion control scheme is developed in such manner it decreases jitter and improve RBUDP's adaptation to network conditions. LambdaStream is an application-layer library, for two reasons. Firstly, application-layer tool makes development easier and simplifies deployment for testing purposes. Secondly, an application-layer protocol can measure end-to-end conditions as applications actually experience them, allowing the protocol to distinguish packet loss and avoid unnecessarily throttling throughput. The key characteristics of the congestion control in LambdaStream are: it is rate based [57], it uses receiving interval as the primary metric to control the sending rate, it calculates rate decrease/increase at the receiver side during a probing phase, and it maintains a constant sending rate after probing for available bandwidth. LambdaStream uses the receiving interval as a metric because 1) the receiving interval is closely related with the link congestion and the receiver's processing capability; 2) the receiving interval can be used to detect incipient congestion and loss differentiation.

The congestion control is composed of two parts. One part is to distinguish a packet loss and adjusts sending rate accordingly, thus avoiding unnecessarily throttling of the sending rate. Another part is to update the sending rate based on the ratio between the average receiving interval and the sending interval. Incipient congestion leads to a higher ratio, which triggers the protocol to decrease the sending rate. The protocol increases its sending rate if the ratio is close to one and the available bandwidth is greater than zero.

LambdaStream extends the congestion control to encompass an end-to-end scope. It distinguishes packet loss and adjusts the sending rate accordingly. The protocol also applies a ratio sampling approach to detect incipient congestion and combines it with a bandwidth estimation method for *proactively* probing for an appropriate sending rate.

LambdaStream protocol's throughput converges very well for a single flow, even for the initial sending rate in between 1720Mbps or 172Mbps. The protocol manages to maintain the throughput at an almost fixed sending rate, about 950Mbps. The experimental results show that LambdaStream achieves 950Mbps throughput in a 1Gbps channel. It exhibits small application level jitter and reacts smoothly to congestion, which is very suitable for streaming applications and also works well for continuous data streams of varying payloads.

5.6. Group Transport Protocol

Group Transport Protocol (GTP), is a receiver–driven transport protocol that exploits information across multiple flows to manage receiver contention and fairness. The key novel features of GTP include 1) request-response based reliable data transfer model, flow capacity estimation schemes, 2) receiver-oriented flow co-scheduling and max-min fairness rate allocation, and 3) explicit flow transition management.

Group Transport Protocol (GTP) is designed to provide efficient multipoint-to-point data transfer while achieving low loss and max-min fairness among network flows [58]. In a multipoint-to-point transfer pattern, multiple endpoints terminate at a receiver and aggregate a much higher capacity than the receiver can handle. In a sender-oriented scheme (e.g. TCP), this problem is more severe because the high bandwidth-delay product of the network makes it difficult for senders to react to congestion in a timely and accurate manner. To address this problem, GTP employs *receiver-based* flow management, which locates most of the transmission control at the receiver side, close to where packet loss is detected. Moreover, a receiver-controlled rate-based scheme in GTP, where each receiver explicitly tell senders the rate at which they should follow, allow flows to be adjusted as quickly as possible in response to detected packet loss.

In order to support multi-flow management, enable efficient and fair utilization of the receiver capacity, GTP uses a receiver-driven centralized rate allocation scheme. In this approach, receivers actively measure progress (and loss) of each flow, estimate the actual capacity for each flow, and then allocate the available receiver capacity fairly across the flows. Because GTP is receiver-centric rate-based approach, it manages all senders of a receiver, and enables rapid adaptation to flow dynamics by adjusting, when flows join or terminate.

GTP is a *receiver-driven response-request* protocol. As with a range of other experimental data transfer protocols, GTP utilizes light-weight UDP (with additional loss retransmission mechanism) for bulk data transfer and a TCP connection for exchanging control information reliably. The sender side design is simple: send the requested data to receiver at the receiver-specified rate (if that rate can be achieved by sender). Most of the management is at the receiver side, which includes a *Single Flow Controller* and *Single Flow Monitor* for each individual flow, and *Capacity Estimator* and *Max-min Fairness Scheduler* for centralized control across flows.

GTP implements two levels of flow control. For each individual flow, the receiver explicitly controls the sender's transmission rate (by sending rate requests to senders). This allows the flow's rate to be adjusted quickly in response to packet loss (detected at the receiver side). Ideally any efficient rate-based point-to-point flow control scheme could be 'plugged in' and it acts as a centralized scheduler. The scheduler at the receiver manages across multiple flows, dealing with any congestion or contention and performing max-min rate amongst them. The receiver actively measures per- flow throughput, loss rate, and uses it to estimate bandwidth capacity. It then allocates the available receiver capacity (can be limited by resource or the final link) across flows. This allocation is done once for each control interval in Max-min fair manner. Correspondingly, the senders adjust to transmit at the revised rates.

Results got after implementation of GTP shows that for point-to-point single flow case, GTP performs well, like other UDP-based aggressive transport protocols (e.g. RBUDP, SABUL), achieving dramatically higher performance than TCP with low loss rates. In case of multipoint-to-point, GTP still achieves high throughput with 20 to 100 times lower loss rates than other aggressive rate-based protocols. In addition, simulation results show, unlike TCP, which is unfair to flows with different RTT, GTP responses to flow dynamics and converge to max-min fair rate allocation quickly.

GTP outperforms other rate based protocols, for multipoint-to-point data transmission, GTP reduces the packet loss, which are caused by aggressiveness of rate based protocols. GTP focus on Receiver based flow contention management, however detailed rate allocation and fairness among flow are not yet considered. Some works has to be done regarding GTP's performance such as achieving max-min fairness, estimating TCP flow's capability and tuning TCP parameters to achieve target rate [57]. Table 8 presents the results for GTP, compared with other rate based protocols such as RBUDP and SABUL. Table 9 displays the data flow statistics for GTP and compares with other protocols.

Protocol	RBUDP	SABUL	GTP
Initial Rate	User defined	Fixed rate	negotiated by sender and receiver
Multipoint to point	No	No	Yes
Rate Adaptation	Optional	Rate based with delay compensation	Rate estimation and delay compensation
Fairness among flows	Not considered	Some extent	max-min fairness among flows at receiver side

TABLE 8: Comparison of GTP with other Rate Based Protocols

Protocol	RBUDP		SABUL		GTP	
	Parallel	**Multi-flows**	**Parallel**	**Multi-flows**	**Parallel**	**Multi-flows**
Aggregate Rate (Mbps)	931	443	912	811	904	865
Average Loss	2.1%	53.3%	0.1%	8.7%	0.03%	0.06%
System Stability	Yes	No	Yes	No	Yes	Yes
Fairness	Fair	No	Fair	No	Fair	Yes

TABLE 9: Parallel vs. Convergent Flow Statistics for GTP

5.7. GridFTP

A common data transfer protocol for grid would ideally offer all the features currently available from any of the protocols in use. At a minimum, it must offer all of the features that are queried for the types of scientific and engineering applications that are intended to support in the gird. For this the existing FTP standard is selected, by adding some features a common data transfer protocol, 'GridFTP' is developed [59]. GridFTP is used as a data transfer protocol for transferring a large volume of data in grid computing. It adopts parallel data transfer which improves the throughput by creating multiple TCP connections in parallel and automatic negotiation of TCP socket buffer size. GridFTP uses TCP as its transport-level communication protocol [60]. In order to get maximal data transfer throughput, it has to use optimal TCP send and receive socket buffer sizes for the link being used. TCP congestion window never fully opens if the buffer size is too small. If the receiver buffers are too large, TCP flow control breaks, and the sender can overrun the receiver, thereby causing the TCP window to shut. This situation is likely to happen if the sending host is faster than the receiving host. The optimal buffer size is twice the bandwidth delay product (BDP) of the link is *Buffersize = 2 * bandwidth delay.*

The GridFTP is implemented in Globus [61] and uses multiple TCP streams for transferring file. Using multiple TCP streams improves performance because of these reasons: i) aggregate TCP buffer size which is closer to real size and ii) circumvents the congestion control.

Several experiments were done for analyzing GridFTP [62]. According to a technical report [63] globus_url_copy achieved a throughput very close to 95%. The windows size was set to bandwidth*RTT, when more than one TCP streams are used, then the window size was set to windows size * num streams. However, to achieve high throughput, the number of TCP connections has to be optimized according to network condition. Problems persist in the file sizes, when the end points want to transfer lots of small files, and then the throughput is reduced.

Figure 7 and 8 shows the download performance for various FTP clients with single and multithreaded flow. Figure 9 shows the data transfer performance among various FTP clients. The performance of GridFTP [64, 65] depends on the number of connections used in parallel, the best performance is achieved with 4 connections and when more connections are there, it creates too much control overhead.

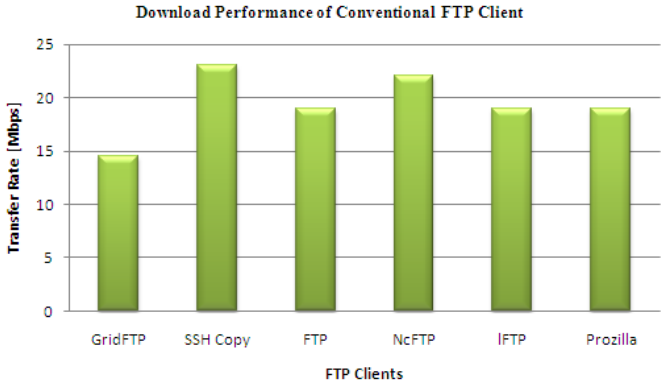


FIGURE 7: Download performance of Conventional FTP Clients [Single Threaded]

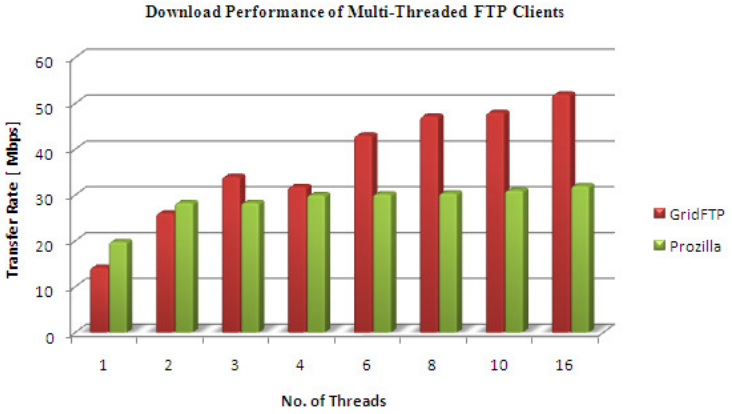


FIGURE 8: Download performance of Multi-threaded FTP Clients

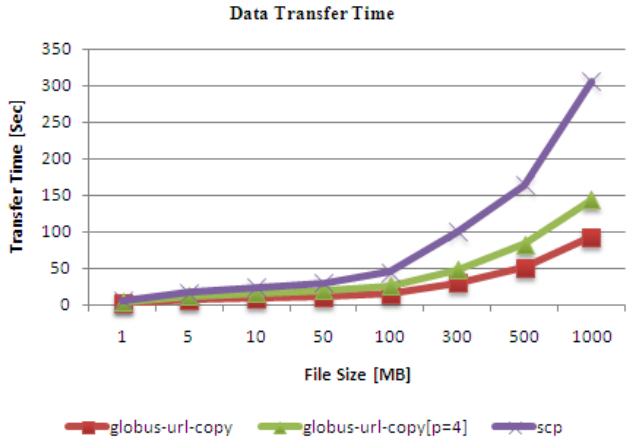


FIGURE 9: Data Transfer Performance of FTP Clients

5.8. GridCopy

GridCopy [66], or GCP, provides a simple user interface to this sophisticated functionality, and takes care of all to get optimal performance for data transfers. GCP accepts scp-style source and destination specifications. If well-connected GridFTP servers can access the source file and/or the destination file, GCP translates the filenames into the corresponding names on the GridFTP servers. In addition to translating the filenames/URLs into GridFTP URLs, GCP adds appropriate protocol parameters such as TCP buffer size and number of parallel streams to attain optimal performance in networks.

Tools such as ping and synack can be used to estimate end-to-end delay; and tools such as IGI [67], abing [68], pathrate [69], and Spruce [70] can be used to estimate end-to-end bandwidth. Latency estimation tools need to be run on one of the two nodes between which the latency needs to be estimated. For data transfers between a client and server, the tools mentioned above can be used to estimate the bandwidth-delay product. However, in Grid environments, users often perform third-party data transfers, in which the client initiates transfers between two servers.

The end-to-end delay and bandwidth estimation tools cited above are not useful for third-party transfers. King [71] developed at the University of Washington at Seattle makes it possible to calculate the round-trip time (RTT) between arbitrary hosts on the Internet. GCP uses King to estimate the RTT between source and destination nodes in a transfer. GCP assumes a fixed one Gbits bandwidth for all source and destination pairs. King estimates RTT between any two hosts in the internet by estimating the RTT between their domain name servers. For example, if King estimates the RTT between the source and the destination to be 50 ms, GCP sets the TCP buffer size to 0.05. GCP caches the source, destination, and buffer size in a configuration file which is available in the home directory of the user running GCP. By default, GCP uses four parallel streams for the first transfer between two sites by a user. GCP calculates the TCP buffer size for each stream by $(BDP/\max(1, streams/l\_f))$, where l_f is set to 2 by default to accommodate for the fact that the streams that are hit by congestion would go slower and the streams that are not hit by congestion would go faster.

The primary design goal for GCP are i) to provide a scp-style interface for high performance, reliable, secure data transfers, ii) to calculate the optimal TCP buffer size and optimal number of parallel TCP streams to maximize throughput and iii)to support configurable URL translations to optimize throughput.

5.9. Summary

Table 10 presents the summary of application layer protocols for high performance grid computing environment.

Protocols	Contributors	Year	Changes Done	Remarks
FOBS	Phillip M. Dickens	2003	Object Based	Less functionality
LOBS	Phillip M. Dickens	2003	Object Based & above FOBS	Lest Functionality
SABUL	Yunhong Gu et. al.	2008	Change in flow control	Application Level Library
UDT	Yunhong Gu et. al.	2009	Rate control + MAIMD	More functionality
Lambda Stream	Chaoyue Xiong et. al.	2005	bandwidth estimation method [proactive]	more bandwidth utilization +small application level jitter
GTP	Ryan Wu et. al.	2004	rate based flow control	Utilizing available bandwidth fastly.
GridFTP	W. Allcock et. al.	2008	Parallel TCP	No User Level Ease
GridCopy	Rajkumar Kettimuthu	2007	Parallel TCP + SCP	User Level Ease

TABLE 10: Summary of Application Layer Protocols for HPC

6. CONCLUSION

A detailed study of the most recent developments on network protocols for grid computing environment is done in this paper. We reviewed the protocols based on TCP and UDP. Below are some points which has to be considered when developing high performance protocols grid computing environment, i) using TCP in another transport protocol should be avoided, ii) using packet delay as indication of congestion can be hazardous to protocol reliability, iii) processing continuous loss is critical to the performance and iv) a knowledge of how much CPU time each part of the protocol costs helps to make an efficient implementation. And also, concentrating on three inter-related research tasks namely: i) dynamic right-sizing, ii) high-performance IP, iii) rate-adjusting, iv) better congestion control and avoidance algorithm and v) efficient flow control algorithm can lead to efficient transport protocol for grid computing environment. Table 11 and 12 gives the comparison chart for TCP based protocols. Table 13 compares the various UDP protocols and Table 14 and 15 gives the application layer protocols comparison.

	STCP	HS-TCP	TCP-Africa	Fast TCP
Design	TCP	TCP	TCP	TCP
Mode of Operation	D2D	D2D	D2D	D2D
Security	No	No	No	No
Congestion control	MIMD	Modified	Modified + 2 Mode CA rule adopted	Depends on parameters
Fairness	No	Yes	Improved	Proportional fairness
TCP Friendly	Yes	Yes	Yes	NA
Performance	Improved	window size and recovery time less	Two mode congestion increases the performance	Well performed
Throughput	Increases	Ok	Good	Increases
Bandwidth Utilization	Ok	Low	Acquires available bandwidth	More
Multicast	No	No	No	No
Implementation details	Linux Kernel 2.4.19	Linux Kernel 2.4.16	NA	NA
Usage	High Speed data Transfer	High Speed data Transfer	High Bandwidth Networks	High Speed data Transfer

TABLE 11: Comparison chart for TCP Based Protocols – Part 1

	TCP-Illinois	CTCP	CUBIC TCP	XTP
Design	TCP	TCP	TCP	TCP
Mode of Operation	D2D	D2D, M2M	D2D	D2D
Security	No	No	No	No
Congestion control	Uses C-AIMD Algorithm and use loss and delay for CC.	Combination of Loss and Delay	Modified	Modified
Fairness	Maintains fairness	Improved	Enhanced	Enhanced
TCP Friendly	Yes	Yes	Yes	Yes
Performance	More than TCP	Increased because of good network utilization	Increased because of good network utilization	Increased, more when used in FDDI, Fibre Channels.
Throughput	Better Throughput	Good	Good	High
Bandwidth Utilization	High	Good	Good	Good
Multicast	No	No	No	Yes
Implementation details	NA	Microsoft and implemented [XP]	NA	Developed by XTP Forum
Usage	High speed TCP variant.	High Speed Data Network.	High Speed Networks	High Speed N/w and Multimedia Applications.

TABLE 12: Comparison chart for TCP Based Protocols – Part 2

	NETBLT	RBUDP	Tsunami
Design	UDP	UDP data + TCP control.	UDP data + TCP control.
Mode of Operation	D2D	D2D & M2M	D2D
Security	No	No	Yes
Congestion control	No	Optional. Limited congestion control can be turned on.	Limited. Sending rate is reduced when loss rate is more than a threshold.
Fairness	No	NA	NA
TCP Friendly	No	No	No
Performance	Performed extremely well	eliminates TCP's slow-start and uses full bandwidth	relays on the parameters
Throughput	Ok	Good	Good
Bandwidth Utilization	Fair	Good	Good
Multicast	No	No	No
Implementation details	No	Provides C++ API.	www.indiana.edu
Usage	developed at MIT for high throughput bulk data transfer	Aggressive protocol designed for dedicated or QoS enabled high bandwidth networks.	Designed for faster transfer of large file over high-speed networks.

TABLE 13: Comparison chart for UDP Based Protocols

	FOBS	LOBS	SABUL	UDT
Design	OO Based	FOBS	UDP data + TCP control.	UDP
Mode of Operation	D2D	D2D	D2D & M2M	D2D & M2M
Security	NA	NA	No	Yes
Congestion control	Yes	Yes	Rate based algorithm	D- AIMD
Fairness	Yes	Yes	Fairness is independent to network delay	Fairness of UDT is independent of RTT
TCP Friendly	Yes	Yes	Yes.	Yes
Performance	90% bandwidth utilized	rate of 35MB per second	Ok	Ok
Throughput	NA	Ok	Ok	Good
Bandwidth Utilization	NA	Good	Ok	High
Multicast	No	No	No	No
Implementation details	NA	NA	C++ Library on Linux.	udt.sourceforge.net
Usage	High Speed data Transfer	High Speed data Transfer	A general purpose protocol.	High Speed data Transfer

TABLE 14: Comparison chart for Application Layer Based Protocols – Part 1

	LambdaStream	GTP	GridFTP	Grid Copy
Design	UDP data + TCP control	Light weight UDP	FTP	FTP
Mode of Operation	D2D	D2D	D2D & M2M	D2D & M2M
Security	No	No	GSI	scp style
Congestion control	Rate Based + Modified	Same as TCP	Same as TCP	Same as TCP
Fairness	Yes	Yes	Yes	Yes
TCP Friendly	Yes	Yes	Yes	Yes
Performance	Improved, because of loss recovery and rate control	Improved because of Receiver driven and flow control	Problems persist in file sizes and number of connections	Problems persist in file sizes (small size)
Throughput	Good	Good	Good	Good
Bandwidth Utilization	Good	Good	High	High
Multicast	No	Yes	Yes	Yes
Implementation details	NA	NA	Globus Toolkit	NA
Usage	Long Fat Networks	High Bandwidth Networks	High Bandwidth networks	High Bandwidth networks

TABLE 15: Comparison chart for Application Layer Based Protocols – Part 2

7. REFERENCES

[1] Foster and C. Kesselman, "*The Grid: Blue print for a new computing infrastructure*", Morgan Kaufmann Publications (1999).

[2] Foster, C. Kesselman, J. M. Nick and S. Tuecke, "*The physiology of the Grid: An open grid services architecture for distributed systems integration*", Grid Forum white paper, 2003.

[3] Volker Sander, "*Networking Issues for Grid Infrastructure*", GFD-I.037, Nov, 22, 2004.

[4] T. DeFanti, C. D. Laat, J. Mambretti, K. Neggers and B. Arnaud, "*TransLight: A global scale LambdaGrid for e-science*", Communications of the ACM, 47(11), November, 2003.

[5] L. Smarr, A. Chien, T. DeFanti, J. Leigh and P. Papadopoulos, "*The OptIPuter*" Communications of the ACM, 47(11), November, 2003.

[6] "*CANARIE*." http://www.canarie.ca/

[7] M. Gaynor, M. Welsh, S. Moulton, A. Rowan, E. LaCombe and J.Wynne, "*Integrating wireless sensor networks with the Grid*", In Proceedings of IEEE Internet Computing, Special Issue on Wireless Grids, July/August, 2004.

[8] J. Postel, "*Transmission control protocol*", RFC 793, September 1981.

[9] J. Postel , "*User datagram protocol*", RFC 768, September 1980.

[10] B. Jacobson, "*TCP extensions for high performance*" ,RFC 1323,May 1992.

[11] Douglas .J. Leith, Robert N. Shorten, "*Next Generation TCP: Open Questions*", In Proceedings of International Workshop on Protocols for Fast Long-Distance Networks, Manchester, UK, 2008.

[12] Ryan X. Wu, Andrew A. Chien et.al., "*A High performance configurable transport protocol for grid computing*", In Proceedings of 5th IEEE International Symposium of Cluster Computing and the grid, Vol.2, pp 1117-1125, 2005.

[13] D. Katabi, M. Hardley, and C. Rohrs, "*Internet Congestion Control for Future High Bandwidth-Delay Product Environments*", ACM SIGCOMM, Pittsburgh, PA, Aug. 19 - 23, pp 89-102, 2002.

[14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "*Modeling TCP throughput: a simple model and its Empirical validation*", ACM Technical Report, UM-CS-1998-008, 1998.

[15] Y. Zhang, E. Yan, and S. K. Dao, "*A measurement of TCP over Long-Delay Network*", In Proceedings of 6th International Conference on Telecommunication Systems, Modeling and Analysis, Nashville, TN, March 1998.

[16] W. Feng et.al., "*The Failure of TCP in High Performance Computational Grids*", Super Computing, ACM/IEEE Conference, November, pp 37, 2000.

[17] R.N. Shorten and D.J. Leith and J. Foy and R. Kilduff, "*Analysis and design of AIMD congestion control algorithms in communication networks*", Article in Automatica, 41(4) pp:725-730, doi:10.1016/j.automatica.2004.09.017, 2005.

[18] Jain, R., Chiu, D.M., and Hawe, W., "*A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems*", DEC Research Report TR-301, 1984.

[19] E. Souza, D. Agarwal, "*A HighSpeed TCP study: characteristics and deployment issues*", Technical Report LBNL-53215, Lawrence Berkeley National Lab, 2003. Available at: http://www-itg.lbl.gov/evandro/hstcp/hstcp-lbnl-53215.pdf.

[20] T.A. Trinh, B. Sonkoly, S. Molnr, "*A HighSpeed TCP study: observations and reevaluation*", In Proceedings of 10th Eunice Summer School and IFIP Workshop on Advances in Fixed and Mobile Networks, Tampere, Finland, 2004.

[21] Tom Kelly, "*Scalable TCP: Improving performance in high speed wide area networks*", ACM SIGCOMM Computing Communications Review, 33(2), April, pp 83-91, 2003.

[22] C. Jin, D.X. Wei, S.H. Low, "*FAST TCP: motivation, architecture, algorithms, Performance"*, In Proceedings of IEEE Infocom , Vol. 4, Hong Kong, China, pp. 2490-2501, 2004.

[23] Gurtov, "*Effect of delays on TCP performance*", In Proceedings of IFIP Personal Wireless Communications 2001 (PWC2001), Lappeenranta, Finland, pp. 810, 2001.

[24] Sumitha Bhandarkar, Saurabh Jain and A. L. Narasimha Reddy, "*LTCP: Improving the Performance of TCP in HighSpeed Networks*", ACM SIGCOMMM Computer Communications Review, 36(1), January, pp 41-50, 2006.

[25] M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo, "*TCP Westwood: Congestion Window Control Using Bandwidth Estimation*", IEEE Globecom 2001, vol: 3, pp 1698-1702, 2001.

[26] Lawrence S. Brakmo, Student Member, IEEE, and Larry L. Peterson, "*TCP Vegas: End to End Congestion Avoidance on a Global Internet*", IEEE JOURNAL ON Selected Areas in Communications, 13(8), October 1995.

[27] M. Fisk and W. Feng, "*Dynamic right-sizing in TCP*", In Proceedings of International Conference on Computer Communication and Networks, October, pp 152-158, 2001.

[28] S. Floyd, S. Ratnasamy, and S. Shenker, "*Modifying TCPs congestion control for high speeds*", Preliminary Draft. URL: http://www.icir. org/floyd/papers/hstcp.pdf, 2002.

[29] S. Floyd, "*HighSpeed TCP for Large Congestion Windows*", RFC 3629, December 2003.

[30] King R, Baraniuk R, Riedi R, "*TCP-Africa: an adaptive and fair rapid increase rule for scalable TCP*", In IEEE Proceedings of IEEE Computer and Communications Societies Conference, vol.3, pp. 1838-1848, doi: 10.1109/INFCOM.2005.1498463, 2005.

[31] Cheng Jin et.al., "*FAST TCP: From Theory to Experiments*", IEEE Network Communications, 19(1), pp 4-11, 2005.

[32] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "*Equation-Based Congestion Control for Unicast Applications*", SIGCOMM 2000

[33] Wei Steven H. Low, Cheng Jin David X, "*FAST TCP: Motivation, Architecture, Algorithms, Performance*", IEEE/ACM Trans Networking, 14(6), pp 1246-1259, 2006.

[34] Shao Liu, Tamer Basar, and R. Srikant, "*TCP-Illinois: a loss and delay based congestion control algorithm for high-speed networks*", In Proceedings of the 1st international conference on Performance evaluation methodologies and tools, ACM, Article 55, doi:10.1145/1190095.1190166, 2006.

[35] Shao Liu, Tamer Basar, and R. Srikant, "*TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks*", Performance Evaluation, 65(6-7) pp.417-440. doi:10.1016/j.peva.2007.12.007, 2008.

[36] Tan, K. Song, J. Zhang, Q. Sridharan, M, "*A Compound TCP Approach for High-Speed and Long Distance Networks*", Proceedings of 25th IEEE International Conference on Computer Communications, pp:1-12, doi:10.1109/INFOCOM.2006.188, 2006.

[37] K. Tan, J. Song, M. Sridharan, and C.Y. Ho,, "*CTCP-TUBE: Improving TCP friendliness over low-buffered network links*", Proceedings of 6th International Workshop on Protocols for FAST Long-Distance Networks, March 2008.

[38] Alberto Blanc, Konstantin Avrachenkov, Denis Collange and Giovanni Neglia, "*Compound TCP with Random Losses*", Springer Berlin / Heidelberg, Lecture Notes in Computer Science, Networking, pp:482-494, doi: 10.1007/978-3-642-01399-7_38, 2009.

[39] Sangtae Ha, Injong Rhee, and Lisong Xu, "*CUBIC: a new TCP-friendly high-speed TCP variant*", SIGOPS Operating Systems Review 42(5) pp: 64-74, doi:10.1145/1400097.1400105, 2008.

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

[41] Diogo R. Viegas, Rodrigo Mario A. R. Dantas, Michael A. Bauer, "SCTP, XTP and TCP as Transport Protocols for High Performance Computing on Multi-cluster Grid Environments", HPCS, pp:230-240, 2009, doi:10.1007/978-3-642-12659-8_17.

[42] Mark L. Lambert, Lixia Zhang, "NETBLT: A Bulk Data Transfer Protocol", RFC 969, December 1985.

[43] David D Clark, Mark L Lamberl, Lixia Zhang, "NETBLT: A High Throughput Transport Protocol", ACM SIGCOMM Computer Communications Review, vol:17, no:5, 1987, pp 353-359, 1987.

[44] Eric He, Jason Leigh, Oliver Yu, Thomas A. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Transfer", Fourth IEEE International Conference on Cluster Computing (CLUSTER'02), pp 317, 2002.

[45] http://dast.nlanr.net/Projects/Iperf/

[46] http://netperf.org/netperf/NetperfPage.html

[47] Mark R. Meiss (2008), "Tsunami: A High-Speed Rate Controlled Protocol for File Transfer", Indiana University, http://steinbeck.ucs.indiana.edu/mmeiss/papers

[48] Altman, E. Barman, D. Tuffin, B. Vojnovic, M, "Parallel TCP Sockets: Simple Model, Throughput and Validation", INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings pp:1 - 12, 2006
doi: 10.1109/INFOCOM.2006.104.

[49] Phillip M. Dickens, "FOBS: A Lightweight Communication Protocol for Grid Computing", Euro-Par 2003 Parallel Processing Lecture Notes in Computer Science, Volume 2790/2003, pp:938-946, 2003. doi: 10.1007/978-3-540-45209-6_130.

[50] www.umcs.maine.edu/~dickens/pubs/LOBS.pdpta.submitted.doc

[51] Yunhong Gu and Robert Grossman, “SABUL: A Transport Protocol for Grid Computing”, Journal of Grid Computing, Vol.1, No.4, December, pp 377-386, 2003.

[52] Phoemphun Oothongsap, Yannis Viniotis, and Mladen Vouk, “Improvements of the SABUL Congestion Control Algorithm”, Proceedings of 1st International Symposium on Communication Systems Networks and Digital Signal Processing, July, 2008.

[53] S. Floyd, M.Mathis, M. Podolsky, “An Extension to the Selective Acknowledgement (SACK) Option or TCP”, RFC 2883, July 2000.

[54] Yunhong Gu and Robert L. Grossman, “UDT: An Application Level Transport Protocol for Grid Computing”, Second International Workshop on Protocols for Fast Long-Distance Networks, PFLDNet, Feb 16-17, 2004, Argonne, Illinois, USA

[55] Yunhong Gu and Robert L. Grossman, “UDT: UDP-based Data Transfer for High-Speed Wide Area Networks”, International Journal of Computer & Telecommunications Networks, Vol.51, No.7, May, 2007.

[56] Chaoyue Xiong, Jason Leigh, Eric He, Venkatram Vishwanath, Tadao Murata, Luc Renambot, Thomas A. DeFanti, "LambdaStream: A Data Transport Protocol for Network-Intensive Streaming Applications over Photonic Networks", In Proceedings of the Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2005), Lyons, France, February 3-4, 2005.

[57] Xinran (Ryan) Wu and Andrew A. Chien, "Evaluation of Rate-Based Transport Protocols for Lambda-Grids", Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC '04). IEEE Computer Society, pp:87-96, 2004, doi:10.1109/HPDC.2004.13.

[58] Wu, R.X., Chien, A.A.,"GTP: Group Transport Protocol for Lambda-Grids", IEEE International Symposium on Cluster Computing and the Grid, (CCGrid 2004), pp: 228- 238, 19-22 April 2004, doi: 10.1109/CCGrid.2004.1336572.

[59] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak (2003), “GridFTP: Protocol Extensions to FTP for the Grid”, GFD-20, April 2003.

[60] John Bresnahan, Michael Link, Gaurav Khanna, Zulfikar Imani, Rajkumar Kettimuthu and Ian Foster, “Globus GridFTP: Whats New in 2007”, Proceedings of International Conference on Networks for grid applications, Article 19, October, 2007.

[61] www.globus.org/toolkit/data/gridftp/

[62] John Bresnahan, Michael Link, Rajkumar Kettimuthu, Dan Fraser and Ian Foster, “GridFTP Pipelining”, Proceedings of the 2007 TeraGrid Conference, June, 2007.

[63] http:\\www.globus.org/toolkit/docs/latest-stable/data/gridftp/gridftp_performance.doc

[64] Hiroyuki Ohsaki, Makoto Imase, "Performance Evaluation of Data Transfer Protocol GridFTP for Grid Computing", International Journal of Applied Mathematics and Computer Sciences 3(1), pp:39-44, 2009.

[65] M. Cannataro, C. Mastroianni, D. Talia, and P. Trunfio, "Evaluating and Enhancing the Use of the GridFTP Protocol for Efficient Data Transfer on the Grid", in Proc. PVM/MPI, pp.619-628, 2003.

[66] Kettimuthu, R, Allcock, W, Liming, L, Navarro, J.-P, Foster, I., "GridCopy: Moving Data Fast on the Grid", IEEE International Parallel and Distributed Processing Symposium, IPDPS 2007. pp:1 - 6, doi:10.1109/IPDPS.2007.370553

[67] http://www.cs.cmu.edu/_hnn/igi/

[68] www.wcisd.hpc.mil/tools/

[69] http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/bw-est/

[70] www.icir.org/models/tools.html/

[71] K. P. Gummadi, S. Saroiu, S., and S. D. Gribble, “King: Estimating latency between arbitrary internet end hosts”, SIGCOMM Computer Communication Review, vol. 32, no. 3, pp. 518, July 2002.