GTP: Group Transport Protocol for Lambda-Grids

Ryan X. Wu and Andrew A. Chien
Department of Computer Science and Engineering
University of California, San Diego
{xwu, achien}@ucsd.edu

Abstract

The notion of lambda-Grids posits plentiful collections of computing and storage resources richly interconnected by dedicated dense wavelength division multiplexing (DWDM) optical paths. In lambda-Grids, the DWDM links form a network with plentiful bandwidth, pushing contention and sharing bottlenecks to the end systems (or their network links) and motivating the Group Transport Protocol (GTP). GTP features request-response data transfer model, rate-based explicit flow control, and more importantly, receiver-centric max-min fair rate allocation across multiple flows to support multipoint-to-point data movement. Our studies show that GTP performs as well as other UDP based aggressive transport protocols (e.g. RBUDP, SABUL) for single flows, and when converging flows (from multiple senders to one receiver) are introduced, GTP achieves both high throughput and much lower loss rates than others. This superior performance is due to new techniques in GTP for managing end system contention.

1. Introduction

Geometric increases in semiconductor chip capacity predicted by Moore's Law have produced a revolution in computing over the past 40 years. Even more rapid advances in optical networking are producing even greater bandwidth increases. The OptIPuter project [1] and other efforts such as CANARIE [2] are exploring the implications of these new "lambda-grid" environments (low-cost, plentiful wide-area bandwidth, plentiful storage and computing) that this revolution enables. The efforts described here are a part of the OptIPuter project.

Circuit-switched lambda's can provide transparent end-to-end optical light paths – available at low cost and delivering huge dedicated bandwidth. Networks of such connections form a lambda-grid (sometimes called a distributed virtual computer) in which the geographically distributed elements can be tightly-coupled. Compared to shared, packet-switched IP networks, lambda-grids have fewer endpoints (e.g. 10^3, not 10^8), dedicated high speed links (1Gig, 10Gig, etc.) between endpoints, which produce an environment within the lambda-grid with no internal network congestion but significant endpoint congestion. In addition, because lambda-grids are likely to connect small numbers of closely interacting resources, our perspective has evolved from a point-to-point model (e.g. data transfer from single server to a client) to a collection of endpoints which engage in multi-point to point, multipoint-to-multipoint communication patterns at high speed. For example, a distributed scientific computation in a data grid might engage in coordinated communication across a number of data servers (endpoints in a group), which fetches large quantities of data (e.g. 10GB) from ten distinct servers to feed a local computation or visualization. These and other similar scenarios pose a new set of research topics for data communication in lambda-grids.

Delivering communication performance in high bandwidth-delay product networks is a long-standing research challenge for just point to point data transfer. Traditional TCP and its variants (e.g. [3] [4], [5]) were developed for shared networks in which the bandwidth on internal links is a critical and limited resource. As such, the congestion control techniques manage internal network contention, providing a reasonable balance of non-aggressive competition and end-to-end performance. As a result, slow-start causes TCP to take a long time to reach full bandwidth when RTT is large, and to recover from packet loss because of its AIMD control law. Figure 1 shows the throughput of TCP varying for 100MB data transmission across a 1Gbps link for different round trip delays.

We focus on the challenge of achieving high performance in complex network structures and communication patterns in lambda-grids. First, with a network setting where internal network congestion is rare, the focus of rate and congestion control shifts to end points (or their access links), and the need to manage it is critical. Second, for multipoint-to-point communication patterns, where flows may have various bandwidth and delay, it is important to achieve a fair rate allocation among flows.

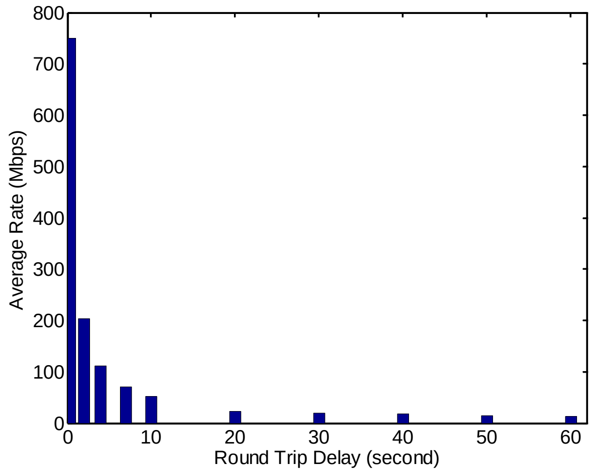

Figure 1: TCP throughput of transferring 100MB data under different round trip delay. Dummynet [6] is used to simulate link delay from 0.5 ms to 60ms in this measurement.

These two key issues motivate our work on the Group Transport Protocol (GTP), 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[7] rate allocation, and 3) explicit flow transition management. Measurements from an implementation of GTP show that for point-to-point single flow case, GTP performs as well as other UDP-based aggressive transport protocols (e.g. RBUDP[8], SABUL[9]), achieving dramatically higher performance than TCP, with low loss rates. Results also show that for multipoint-to-point case, GTP still achieves high throughput with 20 to 100 times lower loss rates than other aggressive rate-based protocols. In addition, simulation results show that unlike TCP, which is unfair to flows with different RTT, GTP responses to flow dynamics and converges to max-min fair rate-allocation quickly.

The remainder of the paper is organized as follows. We describe the multipoint-to-point communication problem of lambda-grids in Section 2. We provide an overview of GTP in Section 3, and present the details of rate and flow control mechanisms of GTP in Section 4. In Section 5 we illustrate the performance of GTP through both ns-2 simulations and real implementation measurements, followed by a summary and discussion of future work.

2. The Problem

2.1 Modeling Lambda-Grid Communications

Dense wavelength division multiplexing (DWDM) allows optical fibers to carry hundreds of wavelengths of 2.5 to 10 Gbps each for a total of terabits per second capacity per fiber. A lambda-grid is a set of distributed resources directly connected with DWDM links (see Figure 2), in which network bandwidth is no longer the key performance limiter to communication. Compared to shared, packet-switched IP networks, the key distinguishing characteristics of lambda-grid networks are:

- Very high speed (1Gig, 10Gig, etc.) dedicated links using one or multiple lambdas (through optical packet-switching or multiple optical network interfaces) connecting a small numbers of endpoints (e.g. 10^3, not 10^8), and possibly long delays (e.g. 60ms RTT from SDSC to NCSA) between sites.
- End-to-end network bandwidth which matches or exceeds the capabilities of the data processing (computing/I/O processing) speeds of attached systems. For example, this supports a model employing geographically distributed storage, allowing fetching data from multiple remote storage sites to feed real-time, local data-computation needs.
- Internal network capacities which support private end-to-end connections for a large number of lambda-grids simultaneously, enabling numerous lambda-grids to have the high speed dedicated bandwidth described above. We can then view the optical links between end system pairs as virtual dedicated connections, which is in contrast to commonly shared links in traditional IP network.

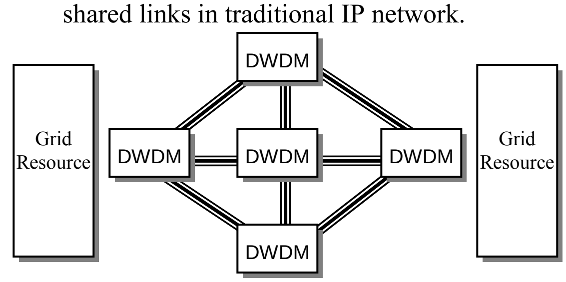

Figure 2: Lambda Grids

We model the set of end-to-end connections and resources which form a lambda-grid as follows. Lambda-grid endpoints and underlying wavelengths form a connection graph, the vertices of which represent distributed sites on lambda grids, and edges denote dedicated links between endpoints. Each edge has a capacity (bandwidth), which is the allocated wavelength capacity between two destinations; that capacity is dedicated to the lambda-grid (not shared with others), This is a different model from shared IP networks, where two end points may be connected through intermediate nodes (e.g. routers) and shared links. We illustrate this in Figure 3, which shows that internal network contention is likely to be shifted from internal network (see Figure 3a) to endpoints (or their access links, where multiple dedicated connections are terminated (see Figure 3b)).

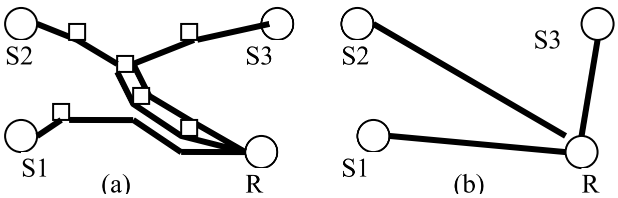

Figure 3: Receiver R's connection's view with three senders. (a) Shared IP connection: senders connect with receiver via shared links and intermediate nodes. (b) Dedicated lambda connection: dedicated capacity between each sender/receiver pair.

2.2 Shift of Management from Network to End System

Recently, communication patterns of lambda-grids have evolved from point-to-point dedicated connection model to a more sophisticated direction where multipoint to point and multipoint to multipoint communications happen. High-speed dedicated wavelength connections will be used to access large distributed data collections (petabytes of online storage), increasing the need to fetch data from multiple sites concurrently to support local computation. This model is already existent in Content Delivery Networks (CDNs) such as Kazaa [10] and BitTorrent [11], where data is stored on multiple replicated servers and clients access multiple servers simultaneously to obtain the desired data. In such a multipoint-to-point setting, where multiple dedicated lambda connections come together, the aggregate capacity of multiple connections is far greater than the data handling speed of the end system. As a result, the critical contention occurs at endpoints, not within the network. This motivates a fundamental change in focus from transport protocols to management of congestion at endpoints.

2.3 Multipoint-to-point Communication Challenges

We formulate the multipoint-to-point group communication problem as follows. Suppose that there are N senders and one receiver on the lambda grid. Let C_r denote the receiver access bandwidth and x_i be the throughput of the connection between sender i ($1 \leq i \leq N$) and receiver. The goal is to maximize aggregate throughput (for the best case, C_r) from these N connections while achieving low loss rate and fairness across flows (with various RTT). Besides sharing same challenges of point-to-point high delay-bandwidth product transmission, we identify the following research challenges around this multipoint-to-point communication problem:

<u>Efficient low loss transmission</u> The solution should be aggressive enough to utilize all of the receiver's communication capacity with multiple connections in a short time period (e.g. several RTT's), while maintaining low average loss rate.

<u>Convergence property</u> If all N flows are long-lived, the rate allocation vector ($x_1, x_2, \ldots, x_N$) should converge to a fixed rate allocation ($x_1^*, x_2^*, \ldots, x_N^*$) regardless of the initial state, flow arrival sequence, or other temporal details.

<u>Fairness among flows</u> The allocation of rates across flows should meet a range of fairness criteria, especially fairness for flows with different RTT values.

<u>Quick reaction to flow dynamics</u> The solution should quickly react flow joins and terminations, as well as converge to a fair allocation from a transition state.

3. GTP Overview

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. In this section we give an overview of GTP, including design rationale, framework, and major features.

3.1 Design Rationale

As described in Section II, lambda-grids shift traffic management from network internal links to end systems. This is especially true for multipoint-to-point transfer pattern, where multiple wavelengths terminating at a receiver, aggregating 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 and happening in lambda-grids because of end-point congestion. Moreover, a receiver-controlled rate-based scheme in GTP, where each receiver explicitly tells senders the rate at which they should follow, allows 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's receiver-centric rate-based approach can manage all senders of a receiver, it enables rapid adaptation to flow dynamics, adjusting seamlessly when flows join or terminate. We describe the features of GTP in more detail in the following subsections.

3.2 Protocol Framework

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, including a *Single Flow Controller* (SFC) and *Single Flow Monitor* (SFM) for each individual flow, and *Capacity Estimator* (CE) and *Max-min Fairness Scheduler* (MFS) for centralized control across flows. The GTP protocol architecture at receiver side is depicted in Figure 4.

3.3 Receiver-oriented Request-Response model

GTP uses two packet types, data and control packets. Each data packet contains a header (including per-connection packet sequence number) and a unit data block (UDB). Control packets are used by receivers to send data/rate requests and exchange information with sender. In the connection setup stage (initiated by either sender or receiver), both ends exchange resource availability, RTT, and sender access bandwidth. The receiver sends data requests and allowed rates to sender via control packets. Within each data request one or a range of UDB's can be requested. The receiver may adjust the sender's rate by sending an updated rate request; however, in a smoothly running system, this should rarely happen.

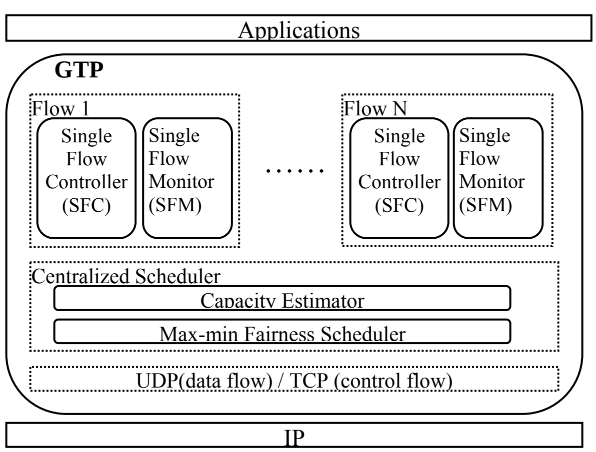

Figure 4: GTP Framework (Receiver)

3.4 Rate Control in GTP

There are two levels of rate control at the receiver: per-flow rate control and centralized rate allocation across flows. At the per-flow level, the *Single Flow Controller* (SFC) manages the sending of data packet requests and chooses the data-request sending rate for each RTT according to measured flow statistics provided by the *Single Flow Monitor* (SFM). The goal is to achieve allocated/target rate by the central scheduler while avoiding congestion. SFC also manages receiver buffer requirements by limiting the number of outstanding UDB requests. At the centralized scheduler level, for each control interval (typically several RTT's), the *Capacity Estimator* (CM) estimates the flow capacity of each individual flow based on the flow statistics provided by SFM. Flow statistics includes the initial allocated rate, achieved transmission rate in the past control interval, packet loss rate, updated RTT estimate, etc. Based on a receiver's set of flow statistics, the central scheduler allocates receiver bandwidth to each flow and updates each flow's target rate. This bandwidth allocation algorithm achieves max-min fairness across flows. The updated allowed (target) rate for each flow is then fed to each SFC. For more detail on flow control and centralized scheduling schemes see Section 4.

3.5 Reliable Transmission

Unlike sender-centric protocols, GTP senders are not responsible for loss retransmission. Lost UDB's are requested again by the receiver. Because packet delivery is expected to be in-order, we employ a per-connection data packet sequence number (embedded in the header of data packet) to diagnose packet loss, as well as calculate transmission and loss rates. If needed, GTP can be augmented to handle out of order delivery.

3.6 Interaction with TCP

As previously mentioned, TCP is not efficient for networks with high delay-bandwidth product links. To perform well on such links, GTP is much more aggressive than TCP. When considering the co-existence of GTP and TCP, there are two mechanisms which support graceful interactions. GTP may adjust its total allocatable bandwidth (distributed to flows by the centralized rate scheduler) to reserve a certain share of the receiver capacity for TCP and other traffic. For instance, GTP may be assigned to utilize up to 80% of the bandwidth and leave 20% for TCP flows. In our future work, we expect to dynamically adjust allocatable GTP traffic by monitoring and estimating TCP traffic, so as to achieve max-min fairness to all flows, including both TCP and GTP.

4. Flow Control and Rate Allocation

In this section we describe the rate control and allocation mechanisms in GTP. In GTP there are two levels of flow control. First, SFC conducts per-flow based control and adjusts flow rate for each RTT. The central scheduler reallocates rates to each flow for each centralized control interval (typically a couple times more than the maximum RTT of individual flows). GTP employs a centralized scheduling algorithm to solve this flow rate allocation problem, which we formally define as a max-min fair rate allocation problem with flow capacity estimation constrains. The contribution of our scheduling scheme is two-fold. First, by scheduling across multiple connections, we are able to make efficient utilization of receiver bandwidth while keeping packet loss low. Second, this guarantees max-min fairness among flows and achieves system convergence. The necessary notations are defined in Table 1.

N	Total number of flows
C_r	Receiver access link capacity
T	Centralized control interval
$R_{i,App}$	Expected (Satisfactory) flow rate specified by application
$R_{i,Send}$	Max. possible bandwidth allocated by sender for flow i
$r_{i,target}$	Target rate (Set by centralized scheduler) of flow i
$r_{i,req}$	Requested rate by receiver for flow i
$r_{i,curr}$	Observed received rate by receiver of flow i over T
$r_{i,loss}$	Observed loss rate by receiver of flow i over T
$r_{i,total}$	Observed total rate of flow i over T, $r_{i,total} = r_{i,curr} + r_{i,loss}$
$loss_i$	Loss ratio of flow i: $loss_i = r_{i,loss} / r_{i,total}$
C_i	Link capability of each flow
RTT_i	Round trip time of flow i
RTT_{max}	Maximum RTT among all the flows.
$\hat{r}_i$	Estimated capacity of flow i
$\Delta \hat{r}_i$	Allowable increment of $\hat{r}_i$

Table 1: Notations

4.1 Single Flow Controller (SFC)

The SFC has two functions: First, it provides per-flow based data packet request management and limits the number of outstanding data requests. This limits the usage of receiver buffers for each flow and prevents receiver flooding when there is congestion. Second, SFC provides per-flow rate adaptation in response to the loss rate. It updates the flow rate and sends the new rate request to the sender every RTT. This enables response to any congestion while trying to achieve allocated rate set by central scheduler. SFC uses a *loss proportional-decrease and proportional-increase* scheme for rate adaptation, which works as follows. For each RTT, based on the current flow loss rate $loss_i$, SFC decreases the requested rate proportional to this loss rate. We also set an upper bound (12.5%) for the decrease in rate.

If there is no packet loss, SFC proportionally increases the requested rate with a small step size (2% per RTT). However the new rate should be no more than the allocated/target rate set by central scheduler. We define per-flow rate update rule as follows:

$$r_{i,req} = \begin{cases} r_{i,req} \cdot (1 - \min\{0.5 \cdot loss_i, 0.125\}) & if \quad loss_i > 0; \\ \min\{r_{i,req}(1+0.02), r_{i,target}\} & if \quad loss_i = 0. \end{cases}$$

We are also exploring efficient delay based single flow rate control mechanism, which will be reflected in our future work.

4.2 Flow Capability Estimator (CE)

The *Capacity Estimator* (CE) provides estimation to the achievable transmission rate of each flow based on its history provided by SFM. At the end of each centralized control interval (by default, we set it as three RTT_{max}), the CE estimates the capacity of each flow, which is used as the upper-bound for that flow during centralized rate allocation phase. Desired rate estimation scheme needs to have the following two characteristics. When there is continuously no packet loss and the achieved throughput is close to target rate, the CE needs to increase the flow's estimate faster. When flow incurs a packet loss, the estimated rate should be reduced according to the loss ratio. We use an *Exponential Increment and Loss Proportional Decrement (EIPD)* scheme for estimation. The idea behind increment/decrement adjustment ($\Delta\hat{r}_i$) is shown in Figure 5. We describe this scheme as follows.

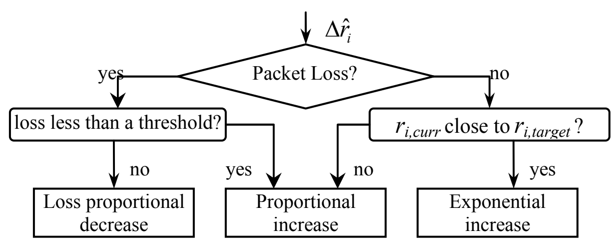

Figure 5: Flow Rate Estimation Scheme

First, when there is no packet loss and the achieved rate is close to previous estimate, we increase the allowable increment of estimation exponentially:

If $r_{i,loss} = 0$ and $r_{i,total} \geq 0.95\ r_{i,target}$, then

If ($\Delta\hat{r}_i = 0$) $\Delta\hat{r}_i = 0.02 \times r_{i,total}$; else $\Delta\hat{r}_i = 2\,\Delta\hat{r}_i$.

When there is no packet loss but the achieved rate is not close to previous estimate, we increase the allowable increment of estimation proportionally:

If $r_{i,loss} = 0$ and $r_{i,total} < 0.95\ r_{i,target}$, then

$\Delta\hat{r}_i = 0.02\ r_{i,total}$.

When there is packet loss, but below certain threshold (0.5%), we still increase the estimate:

If $0 < r_{i,loss} < 0.005$, then

$\Delta\hat{r}_i = 0.02\ r_{i,total}$.

Otherwise (loss ratio is higher than 0.5%), we reduce the estimate proportionally to the packet loss, which is bounded by 20%. Note that $\Delta\hat{r}_i$ is negative in this case:

If $r_{i,loss} > 0.005$, then

$\Delta\hat{r}_i = 0 - \min\{0.5\ r_{i,loss}, 0.2\} \times r_{i,total}$.

After obtaining the allowable increment of flow rate estimation ($\Delta \hat{r}_i$), we set the target rate of each flow as follows:

$$\hat{r}_i = \min \{r_{i,total} + \Delta \hat{r}_i, C_i, R_{i,App}, R_{i,Send}\},$$

where C_i is the link capacity of flow i, $R_{i,App}$ is the expected bandwidth requirement specified by the application and $R_{i,Send}$ is the specified allocatable rate from sender, which is optional. In our future work, senders may be able to explicitly specify the bandwidth $R_{i,Send}$ that is available for flow i.

4.3 Max-min Fair Rate Allocation

For centralized scheduling, an important problem is how to allocate receiver capacity (or its access link bandwidth) to multiple flows (with various RTT) in a fair manner. Max-min fairness [7] is a widely used criterion for bandwidth sharing along single or multiple bottlenecks. For single bottleneck case, under a max-min fair rate allocation, the rate of one flow can be increased only by decreasing the rate of another flow with lower or equal rate. Different from the standard max-min fairness problem, here we need to take into consideration the estimated flow rate, which provides an upper bound for rate allocation. We formally define this fairness criterion as follows:

Definition: Let C_r be of the capacity of all GTP traffic. A rate allocation ($x_i, \ldots, x_N$) under constraint ($\hat{x}_1, \ldots, \hat{x}_N$) is feasible if

$$\sum_{i=1}^{N} x_i \le C_r,$$

and for any i, it holds that $x \le \hat{x}_i$. We call a rate allocation ($x_i, \ldots, x_N$) max-min fair if it is impossible to increase the rate of flow j without losing feasibility or reducing the rate of another flow j' with the rate $x_{j'} < x_j$.

For example, when four flows with different capacities (100, 200, 500, 500) share a link with capacity 1000, the max-min fair rate allocation is (100, 200, 350, 350). This example shows that under max-min fairness, flows with lower achievable rate are given higher priority.

To achieve max-min fairness, we need to allocate bandwidth resource C_r to rate allocation ($r_{i,target}, \ldots, r_{N,target}$) in a max-min fair way with constraint ($\hat{r}_i, \ldots, \hat{r}_N$). We come up with a max-min fair rate allocation scheme, which gives higher priority to flows with lower estimate. To be more specific, our scheme tries to schedule the flow with smaller estimate (than fair share) first, and evenly distribute the remaining bandwidth to those with the estimate higher than average fair share. We describe this scheduling algorithm in Figure 6.

We will show the effectiveness of our scheme by conducting simulations in the next section. However, we leave a formal proof of the convergence properties of our two level rate control mechanism to future work.

We let C_{curr} denote the current available bandwidth, n be the number of flows that have been scheduled.

1 $C_{curr} = C_r$; $n = 0$;
2 $FairShare = C_{curr} / (N-n)$;
3 Find flow j with minimum rate estimate;
4 If $\hat{r}_j < FairShare$ then
5 $r_{j,target} = \hat{r}_j$; $C_{curr} = C_{curr} - r_{j,target}$; n++;
6 Mark flow j as scheduled.
7 Repeat from 2 until $n = N$ or $\hat{r}_j \ge FairShare$
8 Schedule all flows that are not scheduled as:
9 $r_{i,target} = C_{curr} / (N-n)$.

Figure 6: Max-min Rate Allocation Algorithm

4.4 Transition Management.

Transitions happen when new flows join or existing flows terminate. When a new flow starts, we set its estimate to the minimum of its physical link capacity, application specified rate, and sender specified rate (if any):

$$\hat{r}_i = \min \{C_i, R_{i,App}, R_{i,Send}\}.$$

By doing so, we are able to treat new flows just as old ones and apply the same rate allocation algorithm without any changes. When a flow terminates, we try to proportionally increase the estimates of all remaining flows. Let C_l denote the aggregate rates from all the flows that finish within last control interval. Then we increase the capacity estimate of each remaining flow by

$$\hat{r}_i = \hat{r}_i + \frac{\hat{r}_i \cdot C_l}{\sum_j \hat{r}_j}.$$

Again, this allows us to utilize the same centralized scheduling algorithm to conduct rate allocation.

Transitions also happen when we adjust flows rates, and the skew between flows with different RTT's may cause serious end-point congestion. Consider two flows with RTT 5 and 50ms. At the centralized rate allocation stage, we may decrease the rate of flow 1 and increase flow 2 for the sake of achieving max-min fairness. However since flow 2 has a smaller RTT, it responds much faster than flow 1 on rate changes. Also, it may happen that the increased flow 2 arrives at receiver while packets from flow 1 are still arriving at its original higher rate. This causes congestion at the receiver. To solve this problem, we may introduce a delay for rate adjustment to coordinate among flows, and the delay for flow i is defined as

$$Delay_i = RTT_{max} - RTT_i,$$

where RTT_{max} and RTT_i are the maximum round trip time of all the flows, and the round trip time of flow i, respectively. This additional delay will be feedback together with the new target rate update to SFC. Then each single flow controller will delay updating with the new target rate. By doing so we have coordinated the rate update among flows and reduced receiver side congestion caused by the transition of rate changes.

5. Experiments

In this section, we explore the performance of GTP through extensive experiments across three experiment environments. First, we run simulations with the packet-level simulator *ns2*[12] to study the best case performance and packet level dynamics of GTP. *ns2* provides ideal network performance in terms of achievable full link bandwidth, and omitted end system overhead. Second, we utilize a local Dummynet environment (slower end nodes, lossy links with emulated various RTT) to perform a more realistic set of experiments. Finally, we conduct real implementation measurements by comparing GTP with TCP, RBUDP[8] and SABUL[9] on TeraGrid[13] (fixed RTT, fast end-nodes, almost no-loss network links). The multipoint-to-point connection topology is the same as the one in Figure 3b, where multiple senders are connected with receiver (through a gigabit switch) via dedicated links with the same 1Gbps bandwidth but with various delays.

5.1 ns2 Simulation Results

In this subsection we report the *ns2* simulation results of GTP's dynamics to GTP flow changes, its convergence and fairness properties, and its interaction with TCP.

5.1.1 Dynamics of GTP

First, we illustrate the dynamics of GTP by increasing the number of GTP flows arriving at a receiver. Figure 7 shows flow rate trajectories when four flows with different RTT's (20, 40, 60, and 80ms) start at time t =0, 2, 3, 4 seconds. We see that whenever a new flow starts, each active flow achieves an identical fair share of bandwidth. To compare with TCP fairness across flows, we let four TCP flows with the same set of RTT start at time t=0. Figure 8 shows the trajectories of each flow's throughput, from which we see that TCP flow 1 (with shortest RTT) achieves much higher throughput than others. Besides using a max-min fairness criterion, for multiple competing flows with the same link conditions (in this case each flow needs to obtain same rate to achieve fairness), we quantitatively characterize the long-term fairness across multiple flows by using a commonly accepted fairness measurement, defined as:

$$f(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2},$$

where the value of f is between 0 and 1, and x_i is the throughput of flow i. The higher the index f, the fairer the rate allocation is. For GTP flows in Figure 7a, long term (after flow 4 starts) fairness index is 1. And the fairness index of TCP flows in Figure 8 is only 0.67 (throughput ratio of four TCP flows is approximately 6:2:2:1).

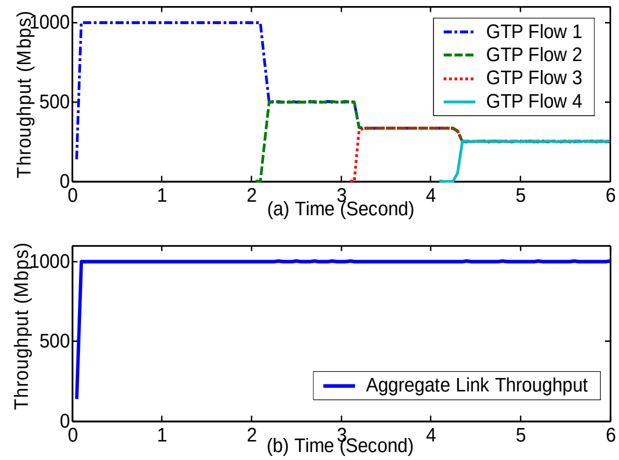

Figure 7: Fairness and convergence of GTP in a multipoint-to-point setting. Four GTP flows are with RTT 20, 40, 60 and 80ms starting at time 0, 2, 3, and 4s.

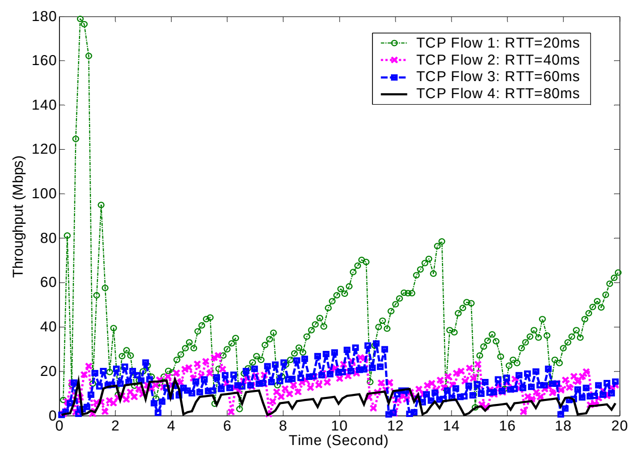

Figure 8: Unfairness over four TCP flows with different RTT (20,40,60 and 80ms), all starting at time t = 0.

We now demonstrate the ability of GTP to probe remaining bandwidth share while achieving max-min fairness. We let GTP flow 1 (with 20ms RTT) starts at time t=0, and GTP flow 2 starts after 2 seconds. Flow 2 is not able to reach the allocated fair share (500Mbps), and only achieves 200Mbps, which may occur if the sender is the bottleneck (e.g. slow disk I/O, or sender/server serves multiple receivers at the same time). From Figure 9 we see that flow 2 remains at 200Mbps while flow 1 increases quickly and reaches 700Mbps, which is the remaining bandwidth. This rate allocation is also max-min fair.

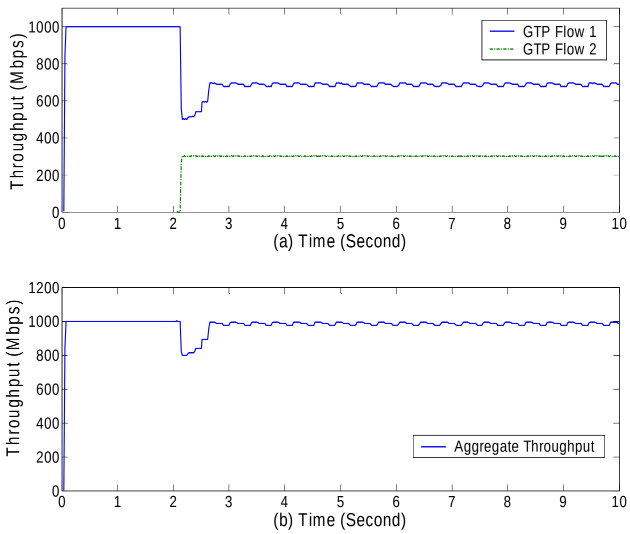

Figure 9: GTP's ability of fast probing the available bandwidth. GTP flow 2 starts at time t=2s, and its maximum transmission rate is 300Mbps.

5.1.2 Interaction with TCP

We illustrate the interaction between GTP and TCP by letting three parallel TCP flows compete with one GTP flow. Figure 10a shows the trajectories of single GTP flow's throughput and the aggregate throughput of three TCP flows, in which case GTP capacity upper bound C_r is equal to full receiver access bandwidth (1000Mbps). Since GTP is very aggressive, it does not let most of the TCP traffic through (see Figure 10a). As suggested in Section IIIf, one possible mechanism for bandwidth sharing between GTP and TCP flows is to limit GTP's total allocatable capacity. Figure 10b shows the result when GTP flow capacity is set to 850Mbps, where TCP traffic could make use of the remaining bandwidth. As a future work, we would like to enable GTP centralized scheduler to support dynamic resource estimation and allocation for both TCP and GTP traffic.

5.2 Dummynet Emulation Results

In our local cluster environment, we configured one cluster node as a Dummynet router, which routes packets while inducing various delays for different flows. The maximum achieved throughput measured by *Iperf*[14] on an emulated 1Gbps link with 60ms RTT is 954Mbps, with 0.3% packet loss. The relatively high packet loss is due to the processing limits of the Dummynet router and the end nodes on our 2Ghz Xeon machines.

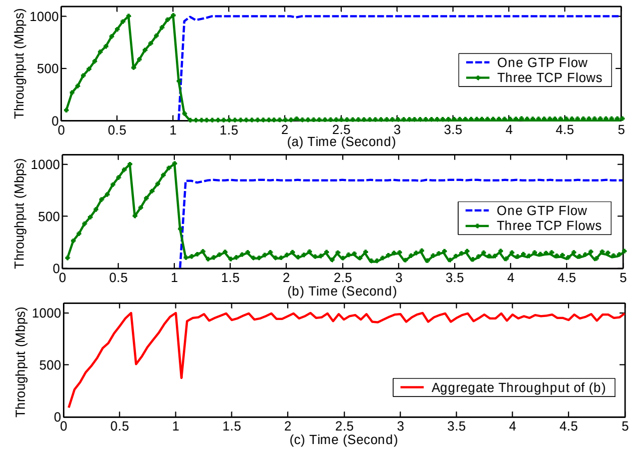

Figure 10: Interaction between GTP and TCP. (a) Three parallel TCP flows (with 1ms RTT) start at t=0, and a GTP flow joins after 1 second with pre-set maximum bandwidth (1Gbps). (b) The case where GTP's maximum bandwidth utilization is set to 85%. (c) The aggregate throughput of all TCP and GTP flows of (b).

We first compare GTP's behavior on Dummynet with the ideal case (provided by the *ns2* simulation) and TCP, in a simple two senders/one receiver case. Flow 1 with 25ms RTT starts at time t=0, and flow 2 with 50ms RTT joins at t=10s. Figure 11 shows the trajectories of these two flows, when they are both either GTP flows on Dummynet, or ideal GTP flows, or TCP flows. We see that for both flows, GTP's performance on Dummynet is close to the ideal case result from ns2 simulation. The fairness index of two GTP flows is 0.99 (from 10s to 33s), while the fairness index of TCP flows is only 0.77.

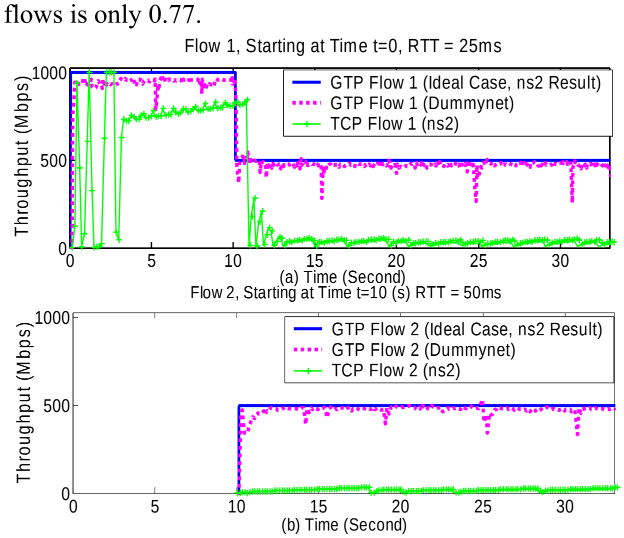

Figure 11: Two flows with different RTT (25 and 50ms) are (1) Idea (simulated) GTP flows (Same as the simulation result in ns2); (2) Real GTP flows on Dummynet; (3) TCP Flows (with tuned large window size).

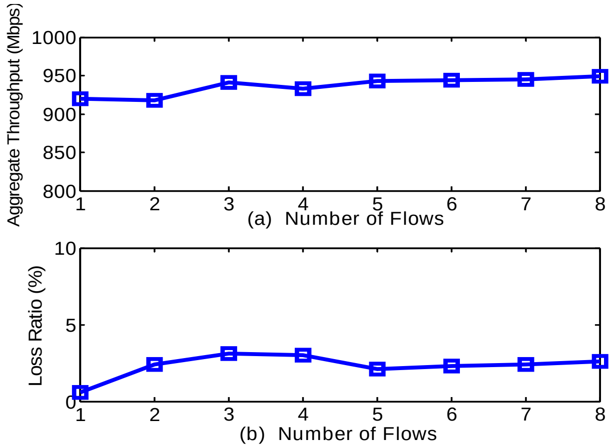

Figure 12: Aggregate throughput and average loss rate of parallel GTP flows. The RTT between sender and receiver is around 25ms.

We now illustrate GTP's performance over different numbers of parallel connections; four cluster nodes are setup as senders and another as a receiver. We vary the number of GTP flows from 1 to 8 (distributed across four senders), and present the results in Figure 12. We see that GTP maintains a high aggregate throughput when the number of parallel flows increases (Figure 12a). We observe high packet loss in the Dummynet environment (compared with Teragrid results in the next subsection), which may be due to the limitations of both the Dummynet router and the end nodes. However we also notice that the loss rate does not always increase with the number of connections, and is bounded by 3%. This loss rate is much lower than other rate based protocols in the same setting, as documented below.

5.3 TeraGrid Experiments: Comparing Rate-based Protocols

5.3.1 Methodology

In this subsection, we compare GTP with two other point-to-point rate based high performance data transfer protocols: RBUDP and SABUL, as well as untuned standard TCP. Reliable Blast UDP (RBUDP) [8] targets reliable data transfer on dedicated or QoS-enabled high speed links. It assumes users have explicit knowledge about the link capacity, requiring the sender to specify an initial rate, start, and maintain its transfer at that rate. SABUL [15] is designed for data-intensive applications in high bandwidth-delay product networks, which starts senders at a fixed high initial rate, adjusting rate based on experienced loss. The newest version of SABUL (UDT) [9] uses delay-based rate adaptation to reduce packet loss caused by its aggressiveness. Throughout our experiments, we use the latest available version of these protocols (RBUDP v0.2, SABUL/UDT v1.0, and GTP prototype). Our experiment is conducted both on Dummynet and TeraGrid[13] (including SDSC and NCSA/UIUC sites). The achievable bandwidth between SDSC and NCSA on each connection is 1Gbps (NIC speed limit). The following performance metrics are used in our experiments:

- Sustained throughput on a 10GB transfer (Point to point and multi-point to point) and average loss rate;
- Fairness for multi-point to point transmission;
- Rate allocation convergence property.

5.3.2 TeraGrid Results

Scenario 1: Point-to-point: Transfer 10GB data from SDSC to NCSA (1Gbps link with 58ms RTT).

	TCP	RBUDP	SABUL	GTP
Time (s)	1639	9.08	8.90	8.92
Avg. Rate	4.88Mbps	881Mbps	898 Mbps	896Mbps
Loss Rate	unknown[1]	0.07%	0.01%	0.02%

TABLE 2: Point-to-Point, from SDSC to NCSA

Our results show, for single, point-to-point high bandwidth delay product paths, the three rate-based protocols all achieve much higher throughput than traditional TCP while maintaining low loss rate.

Scenario 2: Point-to-point, parallel flows: Transfer 10GB data from SDSC to NCSA on the same 1Gbps link with three parallel connections.

	TCP	RBUDP[2]	SABUL	GTP
Aggregate Rate (Mbps)	14.5	931	912	904
Avg. Loss	unknown	2.1%	0.1%	0.03%
System stability	Yes	Yes	Yes	Yes
Fairness	Fair	Fair	Fair	Fair

TABLE 3: Parallel Flows: SDSC to NCSA

In this scenario, all three rate-based protocols perform well when there are parallel flows between sender and receiver, and they all achieve fairness among flows. RBUDP achieves highest throughput as well as loss rate. While RBUDP and SABUL achieve slightly higher throughput than GTP with their aggressiveness, they do so at the expense of a much higher packet loss rate. The GTP's end-point control scheme achieve high throughput with a low loss rate.

Scenario 3: Multi point, convergent flows: Transfer 10GB data; one receiver at SDSC, and three senders with two from NCSA, one from SDSC. Each sender-receiver connection has a 1Gbps dedicated link.

In this case (Table 4), all three rate based protocols achieve high throughput, but loss rates vary over a range of 1000x, with GTP having lowest loss rate by a large margin. GTP also achieves fairness among flows and system stability in this case, while others are not

[1] We are not able to measure instant TCP loss rate, due to the lack of root privileges on TeraGrid.
[2] We assume each flow has no knowledge about others, and starts with the rate of full bandwidth.

fair for flows with different RTT. Note that we assume each RBUDP flow has capacity estimate of 1Gbps, and the loss rate may be reduced if we set each flows' estimated throughput to be less than 1Gbps (see below).

	TCP	RBUDP	SABUL	GTP
Aggregate Rate[3] (Mbps)	677	443	811	865
Avg. Loss	unknown	53.3%	8.7%	0.06%
System stability	Yes	No	No	Yes
Fairness	No	No	No	Yes

TABLE 4: Multi-Point, Convergent Flows

Scenario 4: Two senders and one receiver on Dummynet.

	RBUDP	RBUDP2	SABUL	GTP
Throughput(Mbps)/ Loss Rate of Flow 1	445 27.5%	438 5.2%	326 18.9%	455 2.6%
Throughput(Mbps)/ Loss Rate of Flow 2	256 38.2%	396 5.1%	167 24%	463 3.1%
Aggregate Throughput (Mbps)	701	834	493	918

TABLE 5: Two points to point (with RTT 25 and 50ms), Dummynet. RBUDP2 is the case where we manually set the transmission rate for each RBUDP flow to be half of the link bandwidth.

In this last scenario running on Dummynet, where high packet loss happens, GTP still outperforms other rate based protocols. To summarize the TeraGrid evaluation results, we see that for multipoint-to-point data transmission, GTP significantly reduces the packet loss which is generally caused by the aggressiveness of rate-based protocols.

6. Related Work

The earliest examples of receiver-centric reliable rate based protocols include NETBLT [16] and PROMPT [17]. Recently, the high performance computing community has proposed a range of rate-based point-to-point reliable transport protocols for high bandwidth-delay product networks [8, 9, 15, 18-21]. RBUDP and SABUL are the two with real implementations and measurements on high delay-bandwidth product links. For high delay-bandwidth product links, parallel TCP flows [22, 23] are used to improve the performance of current TCP. However the protocol overhead increases with the increment of the number of parallel connections, and it is not clear about the optimum number of parallel flows. It may also be unfair to other single TCP connections sharing the same bottleneck with these parallel TCP connections.

A key aspect of our research focuses on receiver-based flow contention management. Receiver based multipoint-to-point transmission has been proposed for web traffic [24] and content delivery network [25]. There are some research projects sharing the same idea of receiver based management across flows. In [26], a receiver-side integrated congestion management architecture is proposed, which targets managing traffic across various protocols for real-time traffic. However detailed rate allocation schemes and fairness among flows are not considered. In [27], the authors try to allocate receiver access bandwidth among TCP flows according to their pre-set priority. In the real world, receiver capability may be under-utilized due to the fact that each TCP flow may not be able to always achieve and maintain the allocated rate, and the fairness among flows is not guaranteed due to their priority scheduling scheme. Examples of receiver centric approaches also include [28] for wireless networks. In nearly all cases, these studies focus on networks that are slow relative to the nodes attached to them. In lambda grids, the situation is the opposite.

Another focus of our work is to achieve max-min fairness among flows. Max-min connection fairness has been studied in both the ATM [29, 30] and Internet [31] domains. For fairness criterions other than max-min fairness, we refer to single link fairness index [32] and most recently proportional fairness [33].

7. Summary and Future Work

Recent advances in DWDM networks have fundamentally changed the communication requirements for future lambda grids, where there is sufficient network bandwidth but limited end system capacity. This motivates our work of shifting the network transmission management from the network to the receiver end. We propose GTP, a group transport protocol, as well as a receiver based rate allocation scheme to manage multipoint-to-point transmissions. We design a centralized scheduling algorithm to allocate rate to multiple GTP flows with max-min fairness guarantees. Early results from both ns2 simulation, emulation studies on Dummynet, and real measurements on Terarid show that GTP achieves high throughput, low loss on high bandwidth-delay product links. In addition, results also show that GTP outperforms other point-to-point protocol for multiple-to-point transmission and achieves fast convergence to max-min fair rate allocation across multiple flows.

We identify the following future work. First, we are interested in improving per-flow based control scheme to react more efficiently to irregular network traffic (e.g. bursty traffic). Second, we are studying how to introduce TCP traffic management into our centralized control scheme. This includes open questions such as how to estimate TCP flow's capability, how to

[3] Aggregate rate and loss rate vary for RBUDP and SABUL, and numbers listed are the average values of several measurements.

efficiently tune TCP parameters to achieve target rate, etc. Third, we are working on formal proofs of system convergence properties for our two level flow control schemes. Finally, we expect to integrate GTP into a Distributed Virtual Computer (DVC) [34], a simple grid program execution environment being developed for lambda-grids as part of the OptIPuter project. Within a DVC, GTP will provide high speed communication and data transfer services to applications

Acknowledgements

Supported in part by the National Science Foundation under awards NSF EIA-99-75020 Grads and NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF NGS-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from Hewlett-Packard, BigBangwidth, Microsoft, and Intel is also gratefully acknowledged.

8. References

[1] L. Smarr, A. Chien, T. DeFanti, J. Leigh and P. Papadopoulos, *The OptIPuter.* Communications of the Association for Computing Machinery. **47**(11).

[2] *CANARIE.* http://www.canarie.ca.

[3] V. Jacobson, *Congestion Aviodance and Control.* Computer Communication Review, vol. 18, no. 4, August 1988.

[4] L. Brakmo and L. Peterson, *TCP Vegas: End to End Congestion Avoidance on a Global Internet.* IEEE Journal of Selected Areas in Communications,. **13**(8): p. 1465-1480.

[5] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow., *TCP Selective Acknowledgement Options.* RFC2018, Internet Engineering Task Force (IETF), October 1996.

[6] L. Rizzo, *Dummynet: a Simple Approach to the Evaluation of Network Protocols.* Computer Communication Review, January 1997. **27**.

[7] D.P. Bertsekas and R. Gallager, *Data Networks, Second Edition.* Prentice-Hall, New Jersey, 1992.

[8] E. He, J. Leigh, O. Yu and T. DeFanti, *Reliable Blast UDP: Predictable High Performance Bulk Data Transfer.* IEEE Cluster Computing, 2002: p. 317.

[9] Y. Gu, X. Hong, M. Mazzucco and R.L. Grossman, *SABUL: A High Performance Data Transfer Protocol.* Submitted for publication.

[10] *Kazaa.* http://www.kazaa.com.

[11] *BitTorrent.* http://bitconjurer.org/BitTorrent/.

[12] *Network Simulator - ns2.* http://www.isi.edu/nsnam/ns/.

[13] D.A. Reed, *Grids, the TeraGrid, and Beyond.* IEEE Computer, 2003. **36**(1): p. 62-68.

[14] *Iperf Tool.* http://dast.nlanr.net/Projects/Iperf/.

[15] H. Sivakumar, R. Grossman, M. Mazzucco, Y. Pan and Q. Zhang, *Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks.* to appear in Journal of Supercomputing, 2003.

[16] D.D. Clark, M. Lambert and L. Zhang, *NETBLT: A High Throughput Transport Protocol.* Proceedings of ACM SIGCOMM '88, 1988.

[17] T.S. Balraj and Y. Yemini, *PROMPT - A Destination Oriented Protocol for High Speed Networks.* IFIP WG 6.1/WG 6.4, Palo Alto, CA, Nov. 1990.

[18] P.M. Dickens and W. Gropp, *An Evaluation of a User-Level Data Transfer Mechanism for High Performance Networks.* in Proceedings of the 12th High Performance Distributed Computing (HPDC12) conference, 2003.

[19] *Tsunami.* http://www.indiana.edu/~anml/anmlresearch.html.

[20] A. Feng, A. Kapadia, W. Feng and G. Belford, *Packet Spacing: An Enabling Mechanism for the Delivery of Multimedia Content.*

[21] P. Dickens, *FOBS: A Lightweight Communication Protocol for Grid Computing.* in Proceedings of Euro-Par 2003.

[22] B. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel and S. Tuecke, *Data management and transfer in high-performance com- putational grid environments.* Parallel Computing. **28**(5): p. 749-771.

[23] H. Sivakumar, S. Bailey and R.L. Grossman, *PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks.* Proceedings of Supercomputing 2000.

[24] R. Gupta, M. Chen, S. McCanne and J. Walrand, *WebTP: A Receiver-Driven Web Transport Protocol.*

[25] P. Rodriguez and E.W. Biersack, *Dynamic Parallel Access to Replicated Content in the Internet.* IEEE/ACM Transactions on Networking, 2002. **10**(4): p. 455-465.

[26] H. Balakrishnan, H.S. Rahul and S. Seshan, *An Integrated Congestion Management Architecture for Internet Hosts.* Proceedings of ACM SIGCOMM 1999.

[27] P. Mehra, A. Zakhor and C.D. Vleeschouwer, *Receiver-Driven Bandwidth Sharing for TCP.* in Proceedings of IEEE INFOCOM 2003.

[28] H.-Y. Hsieh, K.-H. Kim, Y. Zhu and R. Sivakumar, *A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces.* Proceedings of the 9th annual international conference on Mobile computing and networking, 2003: p. 1-15.

[29] *Traffic Management Specification.* ATM Forum Version 4.1, af-tm-0121.000, 1999.

[30] A. Charny, D.D. Clark and R. Jain, *Congestion Control with Explicit Rate Indication.* In Proceedings of ICC'95, June 1995.

[31] P. Karbhari, E. Zegura and M. Ammar, *Multipoint-to-Point Session Fairness in the Internet.* in Proceedings of IEEE INFOCOM 2003.

[32] D.-M. Chiu and R. Jain, *Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks.* Journal of Computer Networks and ISDN Systems. **17**(1).

[33] F. Kelly, A. Maulloo and D. Tan, *Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability.* Journal of the Operational Research Society, 49, pp. 237-252.

[34] N. Taesombut and A.A. Chien, *Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications.* to appear in Proceedings of the Workshop on Grids and Advanced Networks (GAN 04), 2004.