FOBS: A Lightweight Communication Protocol for Grid Computing

Phillip M. Dickens

Abstract

The advent of high-performance networks in conjunction with low-cost, powerful computational engines has made possible the development of a new set of technologies termed *computational grids*. These technologies are making possible the creation of very large-scale distributed computing systems by interconnecting geographically distributed computational resources via very high-performance networks. This provides tremendous computational power that can be brought to bear on large-scale problems in several domains, making feasible the development of high-performance distributed applications that, due to the low bandwidth and best-effort nature of the Internet1 environment, were heretofore infeasible.

However, many problems remain before Grid computing can reach its full potential. One particularly difficult issue is that of utilizing fully the available bandwidth while being in some sense fair to competing traffic flows. It has been widely demonstrated that TCP, the communication protocol of choice for most distributed application, often performs quite poorly in the emerging high-bandwidth high-delay network environments. This has led to significant research on the development of user-level applications that can circumvent some of the performance problems inherent in TCP.

In this paper, we discuss our work on developing an efficient, lightweight application-level communication protocol for the high-bandwidth, high-delay network environments typical of computational grids. The goal of this research is to provide congestion-control algorithms that allow the protocol to obtain a large percentage of the underlying bandwidth when it is available, and to be responsive (and eventually proactive) to developing contention for system resources. One advantage of operating at the application level is that a wealth of information related to the behavior and performance of the data transfer (from the perspective of the application itself) can be collected and maintained (a task that is very difficult to perform at the kernel level). We believe that such historical data can be leveraged by the congestion-control mechanism to both improved performance and realize better utilization of system resources. This research represents our initial attempt to develop such algorithms. Towards this end, we develop and evaluate two application-level congestion-control algorithms, one of which incorporates historical knowledge and one that only uses current information. We compare the performance of these two algorithms with respect to each other and with respect to TCP.

1 Introduction

The national computational landscape is undergoing radical changes as a result of the introduction of cutting-edge networking technology and the availability of powerful, low-cost computational engines. This combination of technologies has led to an explosion of advanced high performance distributed applications that, because of the limited bandwidth and best-effort nature of the Internet1 environment, were heretofore infeasible. Concurrently, research efforts have focused on the development of *Grid computing*, a fundamentally new set of technologies that create large-scale distributed computing systems by interconnecting geographically distributed computational resources via very high-performance networks. The advanced applications being developed to execute in Grid environments include distributed collaboration across the Access Grid, remote visualization of terabyte (and larger) scientific data sets, large-scale scientific simulations, Internet telephony, and multimedia applications. The emerging Grid technologies are becoming increasingly important to the national computational infrastructure, and research projects aimed at supporting these new technologies are both critical and timely.

Arguably, Grid computing will reach its vast potential if, *and only if*, the underlying networking infrastructure (both hardware and software) is able to transfer vast quantities of data across (perhaps) quite long distances in a very efficient manner. Experience has shown, however, that advanced distributed applications executing in existing large-scale computational Grids are often able to obtain only a very small fraction of the available underlying bandwidth [8, 13-15, 20, 35, 36] . The reason for such poor

performance is that the Transmission Control Protocol (TCP) [34], the communication mechanism of choice for most distributed applications, was not designed and is not well suited for a high-bandwidth, high-delay network environment [8, 13, 14, 16, 20, 32, 33, 35, 36, 40]. This issue has led to research aimed at improving the performance of the TCP protocol itself in this network environment [1, 4, 18, 28, 31, 32], as well as developing application-level techniques that can circumvent the performance problems inherent within the protocol [14-16, 25, 26, 30, 33, 35, 36]

Despite all of the research activity undertaken in this area, significant problems remain in terms of obtaining available bandwidth while being in some sense fair to competing flows. While TCP is able to detect and respond to network congestion, its very aggressive congestion-control mechanisms result in poor bandwidth utilization even when the network is lightly loaded. User-level protocols such as GridFTP [2], RUDP [26, 30], and previous versions of FOBS [13, 15] are able to obtain a very large percentage of the available bandwidth. However, these approaches rely on the characteristics of the network to provide congestion control (that is, they generally assume there is no contention in the network). Another approach (taken by SABUL [36]) is to provide an application-level congestion-control mechanism that is closely aligned with that of TCP (i.e. using the same control-feedback interval of a single round trip time).

We are attempting to approach the problem from a somewhat different perspective. Rather than attempting to control the behavior of the protocol on a time scale measured in milliseconds, we are interested in developing approaches that can operate on a much larger time scale while still providing very effective congestion control. We believe the best way to achieve this goal is to use the historical information that is available to the application to help drive the congestion-control algorithms. The work presented in this paper is aimed at showing that our approach is both feasible and useful.

The rest of the paper is organized as follows. In Section 2, we provide a brief overview of the components of the FOBS data transfer system. In Section 3, we discuss the design of two user-level congestion control algorithms. In Section 4, we discuss the design of the experiments developed to evaluate the performance of our control mechanisms and to compare their performance with that of TCP. in Section 5, we discuss the results of these experiments. We discuss related work in Section 6, and provide our conclusions and future research directions in Section 7.

2. FOBS

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 acknowledgment and retransmission mechanism. Experimental results have shown that FOBS performs extremely well in a computational Grid environment, consistently obtaining on the order of 90% of the available bandwidth across both short- and long-haul network connections [13-15, 39]. Thus FOBS addresses quite well the issue of obtaining a large percentage of the available bandwidth in a Grid environment. However, FOBS is in-and-of-itself a very aggressive transport mechanism that does not adapt to changes in the state of the end-to-end system. Thus to make FOBS useful in a general Grid environment we must develop congestion control mechanisms that are responsive to changes in system conditions while maintaining the ability to fully leverage the underlying high-bandwidth network environment.

2.1 Rate-Controlled FOBS

Rate-Controlled FOBS (RCF) is implemented on top of the FOBS data transfer engine. There is a simple control agent residing at both communications endpoints that controls the activity of the data transfer engine. Currently, these agents have functionality limited to carrying out the basic congestion control algorithms, but we are currently developing more sophisticated functionality. The data transfer engine itself is (at least conceptually) reasonably simple. While it is beyond the scope of this paper to discuss in detail the implementation of FOBS and the technical issues addressed (the interested reader is directed to [12-15] for such a discussion), it is worthwhile to briefly discuss the implementation of the reliability mechanism.

FOBS employs a simple acknowledgment and retransmission mechanism. The file to be transferred is divided into data units we call *chunks*, and data is read from the disk, transferred to the receiver, and written to disk in units of a chunk. Currently chunks are 100 MBs, this number being chosen based on extensive experimentation. Each chunk is subdivided into *segments*, and the segments are further divided into *packets*. Packets are 1,470 bytes (within the MTU of most transmission mediums), and a segment consists of 10,000 packets. The receiver maintains a bitmap for each segment in the current chunk depicting the received/not-received status of each packet in the segment. A 10,000-packet segment requires a bitmap of 1,250 bytes, which will also fit in a data packet within the MTU of most transmission media. These bitmaps are sent from the data receiver to the data sender at intervals dictated by the protocol, and trigger (at a time determined by the congestion/control flow algorithm) a retransmission of the lost packets. The data to be transferred uses UDP sockets and the bitmaps are sent on TCP sockets.

There are two advantages of using this approach. First, such segmentation of the data de-couples the size of the acknowledgment packet from the size of the chunk. That is, if the bitmaps are pegged to the chunk size, the size of the bitmaps would increase linearly with the size of the chunk. Given a chunk size of 100 MBs, the corresponding bitmaps would be on the order of 8,700 bytes. While the issue can certainly be mitigated to some extent by using negative acknowledgments, it is not difficult to imagine that it may still require bitmaps of a size larger than the MTU. This makes it more difficult to deliver the feedback in a timely fashion (it would have to be fragmented along the way), which could have a very detrimental impact on performance.

Perhaps more importantly, these bitmaps provide a precise and detailed picture of the packet-loss patterns caused by the current state of the end-to-end system. We term these bitmaps *packet-loss signatures*, and preliminary research suggests that as the events that drive packet loss change the packet-loss signatures themselves also change. Currently, the packet-loss signatures are fed to a visualization system as the transfer is taking place, and we are attempting to develop an understanding of these signatures. The goal is to incorporate information gleaned from these bitmaps to help guide the congestion-control mechanisms.

3 Congestion-Control Algorithms

We are interested in evaluating approaches to congestion control that operate under a different set of assumptions than current techniques. One issue in which we are interested is whether the feedback-control interval must be pegged to roundtrip times in order to provide effective congestion control and ensure fairness. This approach is problematic for two reasons: First, a small spike in packet loss may well represent an event that has dissipated by the time the sender is even aware of its occurrence. Thus the congestion control mechanism may be reacting to past events that will not re-occur in the immediate future. Secondly, such an approach can become unstable. That is, the algorithm can get into cycles of increasing the sending-rate due to low packet loss, followed by a spike in packet-loss due to the increased rate, followed by aggressively backing off in reaction to increased loss. For these reasons, we believe it is important to explore alternative approaches.

We have developed and tested two alternative approaches to congestion control, one where the feedback-control interval is significantly lengthened and the increase/decrease parameters for the send-rate are linear. This approach incorporates historical knowledge into the design of the algorithm. The other approach is *state-based*, where the behavior of the algorithm depends only upon the current state and the current feedback.

It is important to note the way in which historical knowledge is incorporated into the first protocol. In particular, the algorithm was developed based on empirical observations made during the development and testing of the FOBS system. Thus the historical patterns were deduced by the researchers, and deducing such information from statistical measures of protocol behavior is of course significantly more complex. However, it does point to the fact that historical information can be important. One observation was that once the protocol found a sending rate that resulted in negligible packet loss, it could remain at that rate for a reasonably long period of time (which in general was longer than it took to successfully transfer one complete chunk of data). Another observation was that once such a rate was established, the best approach was to stay at that rate. That is, there appeared to be an “optimal” sending rate such that being more

aggressive tended to result in non-proportional increases in packet loss, and becoming less aggressive did not result in a meaningful decrease in packet loss. Finally, given a current "optimal" sending rate S, it was observed that the next such sending rate was quite often close to S. Such characteristics were not observed on all connections all the time. However, it was a pattern that emerged frequently enough to be noticed.

3.1 Using Historical Knowledge

These observations were incorporated into the protocol in the following ways. First, the feedback-control interval was extended to the time required to successfully transfer one complete chunk of data. The mean loss rate for an entire chunk was calculated once the data had been successfully transferred, and if this rate exceeded a threshold value the send-rate was decreased. The threshold value was ½ of 1%, and the decrease parameter was a constant 5 Mbs on a 100 Mbs link and 50 Mbs on Gigabit connection. This slow decrease in the send rate is clearly not appropriate when the current conditions are causing massive packet loss *and* a reduction in sending rate (or several reductions) does not have a significant impact on the loss rate. We are investigating techniques to detect such conditions and respond by either raising the decrease parameter or perhaps by switching to another protocol.

The increase parameter for this approach was also linear, but the algorithm does *not necessarily increase* the sending rate each time the loss-rate falls (or remains) below the current threshold value. Rather, the protocol keeps track of the current rate, the number of times the sending rate has been incremented from the current rate, and the number of times such an increase has resulted in a significant increase in packet loss. Based on this sample, the probability that increasing the sending rate will result in increased packet loss is computed, and the protocol then uses this probability to determine if the sending rate will be increased or remain unchanged.

It is important to note that this protocol incorporates two different types of historical knowledge. One type of knowledge was the empirical observations that suggested the basic framework of the algorithm. The other source of knowledge was obtained by tracking the historical relationship between an increase in the sending rate (from a given rate A to A + increase_amount) and an increase in the loss-rate. Thus the protocol captured information obtained both within a given session and between multiple sessions.

3.2 State-Based Approach

We also investigated what may be termed a *state-based* approach, where decisions regarding the send rate are based solely on the *current state* and *current feedback* information. We have developed such an algorithm and have implemented three states thus far: Green, Yellow, and Red. These states represent excellent, moderate, and poor system conditions respectively. Each state has its own minimum and maximum sending rate, its own increase and decrease parameters, and its own set of conditions under which it will change state. The feedback-control interval is the amount of time required to successfully transfer *one segment* of data to increase the responsiveness of the protocol to current conditions. The parameters are set such that the sending rate is incremented at a high rate when there is little packet loss and decremented very quickly in the event of massive packet loss.

4 Experimental Design

Our experiments were conducted on links between Argonne National Laboratory and the Center for Advanced Computing Resource (CACR), and links between Argonne National Laboratory and the National Center for Supercomputing Applications (NCSA). The round trip time between ANL and CACR (as measured by traceroute) was approximately 64 milliseconds, which we loosely categorize as a *long-haul* connection. The round trip time between ANL and NCSA was on the order of eight milliseconds, which we (again loosely) categorize as a *short-haul* connection. All of the links were connected through the Abilene backbone network, and all hosts had a Gigabit Ethernet interface (although the number of routers traversed before reaching Abilene was not clear).

Two hosts were tested within NCSA: one was a 64 processor SGI Origin 2000 running IRIX 6.5 (modi4), and the other was a 64 bit IBM IntelliStation Z Pro 6894 workstation running Linux 2.4.16 (user01).

User01 has an Intel 800MHz Itanium CPU (duel processor), and each processor of the SGI was a 195MHz MIPs R10000. The host at CACR was a four-processor HP N4000 running HP-UX 11. At Argonne National Laboratory, we ran our tests on a dual processor Intel i686 running Linux RedHat 6.1. All experiments were conducted using a single processor.

We simulated the transfer of a 10-Gigabyte file from user01, skinner, and modi4 to terra. That is, we performed the number of iterations necessary to complete a transfer of that size but did not actually perform the read from and write to disk (largely because of disk quotas on some hosts). We were unable to send in the opposite direction because of firewalls at ANL. The tests were conducted during normal business hours (for all hosts involved) to ensure there would be some contention for network and/or CPU resources. The metrics of interest were the mean throughput and the overall loss percentage[1].

We were also interested in comparing the performance of our approach with that of TCP, and did conduct experiments between these same links to capture this information. However, the results obtained with TCP were extremely poor due to the small TCP buffer size on terra (64 KB) that could only be changed with root permissions (which we did not have). These results will be provided, but we will also discuss results obtained in previous experiments[13-15] where the Large Window extensions for TCP were enabled.

5 Experimental Results

The results of these experiments are shown in Tables 1 – 3. The major difference between the two user-level algorithms is the higher loss rate experienced by the state-based approach on the link between ANL and CACR. This larger loss rate was a reflection the instability of the state-based approach on that particular link when contention was present. The instability arose from the fact that the algorithm immediately increases the sending rate whenever the loss rate fell (or remained) below the threshold value. Thus it spent a significant amount of time oscillating between sending at the "optimal" rate (as discussed in Section 3), and a rate that was not significantly higher but that resulted in significantly increased packet loss nonetheless. This phenomenon was due to the fact that negligible packet loss was experienced on the short-haul connections, and the algorithm thus rarely had to decrease the send rate.

Another interesting result was that FOBS was not able to obtain a very large percentage of a gigabit connection across any of the connections. There were two primary factors contributing to this relatively low utilization of network bandwidth. First, the communication endpoints were all shared by many users that significantly reduced the number of CPU cycles dedicated to the transfer (at both the sending and receiving endpoints). The throughput was much higher (on the order of 390 Mbs between ANL and the hosts at NCSA) when the experiments were run late at night with little competition for CPU (or network) cycles.

The second factor has to do with inefficiencies of the FOBS implementation. Currently, the **select()** statement is used to determine those sockets ready for either sending or receiving. This is a very heavyweight system call that is used repeatedly in the current implementation. A better approach would be to use threads to determine when a socket becomes available, and we are currently implementing a multi-threaded version of FOBS.

It is also quite clear that the application-level approach obtained performance significantly higher than that obtained by TCP across all experiments with very little packet loss. As noted above, the primary reason for TCP's very poor performance has to do with the very small TCP buffers available on terra (which has a significant negative impact on TCP's performance). In previous experiments that we conducted [13-15], using the same links but different hosts (that supported the Large Window extensions for TCP [28]), the results were better. In particular, on the link between ANL and NCSA TCP obtained on the order of 90 Mbs, and obtained on the order of 50 Mbs on the link between ANL and CACR. However, even with support for large buffer sizes, TCP's performance was significantly less than that obtained by FOBS.

[1] The loss percentage was calculated as follows. Let N be the total number of packets that had to be sent (assuming no packet loss), and let T be the total number of packets actually sent. Then the loss rate $R = (T - N) / N$.

6 Related Work

The work most closely related to our own are RUDP [26, 30] and SABUL[36], both of which have been developed at the Electronics Visualization Laboratory at the University of Illinois Chicago. RUDP is designed for dedicated networks (or networks with guaranteed bandwidth), making the issue of congestion control largely irrelevant. However, since FOBS is designed to operate in a general Grid environment the issues of congestion control must be addressed. SABUL does provide congestion control that uses a feedback-control interval pegged to round trip times. We believe this interval is too small for large-scale data transfers across high-performance networks, and the research presented in this paper is an attempt to move away from such a model of congestion control.

Research related to application-level scheduling and adaptive applications is also related (e.g. the Apples project [3, 6, 7, 9, 17, 19, 37, 38] and the Grads project [5, 10, 11, 21, 23, 29]). This body of research is focused on interactions between an application scheduler and a heavyweight Grid scheduler provided in systems such as Globus [22] and Legion [24, 27]. FOBS on the other hand is a very lightweight communication protocol that operates outside of the domain of such large systems. However, an interesting extension of our work would be to modify FOBS such that it could interact with Grid-level schedulers to negotiate access to system resources.

7 Conclusions and Future Research

In this paper, we described a lightweight application-level communication protocol for computational Grids. The protocol is UDP-based, with a simple acknowledgment and re-transmission mechanism. We discussed two different application-level congestion-control mechanisms, and showed that Rate-controlled FOBS was able to significantly outperform TCP, across all connections tested, with very minimal packet loss (especially when historical knowledge was incorporated into the algorithm).

There are many ways in which this research can be extended. The results provide herein suggest that integrating historical knowledge into the congestion control mechanism can significantly enhance its performance. This of course brings up the issue of how to automate the collection of historical data, which data has the best chance of providing insight into future behavior, and how such data can be incorporated into the control mechanisms. These are all issues we are currently pursuing.

Also, it is our goal to make FOBS a widely used transport protocol within computational Grids, and we are therefore interested in making it more efficient, providing a sophisticated front end, and further developing the visualization system. The visualization system is, in our view, important because it provides the user with real-time feedback related to the state of the data transfer as well as pictures of the packet-loss signatures. The goal is to use such information to develop a deeper understanding of the interaction between the communication protocol and its surrounding execution environment.

To maximize the performance of FOBS on gigabit connections we are developing a multi-threaded version of the protocol. Also, we are working on an implementation of FOBS for a cluster environment where the sender and receiver are both implemented across multiple CPUs.

	Throughput	Packet-loss Percentage
Historical Information	190 Mbs	Negligible
Current state information	192 Mbs	Negligible
TCP	7.2 Mbs	----

	Throughput	Packet-loss Percentage
Historical Information	86 Mbs	0.59 %
Current state information		
TCP		

	84 Mbs	2.4 %
	1.6 Mbs	----

Table 1. This table shows the performance of the different protocols on the link between ANL and user01 at NCSA.

Table 2. This table shows the performance of the different protocols on the link between ANL and CACR.

	Throughput	Packet-loss Percentage
Historical Information	161 Mbs	Negligible
Current state information	159 Mbs	Negligible
TCP	6.5 Mbs	----

Table 3. This table shows the performance of the different protocols on the link between ANL and modi4 at NCSA.

References

[1] *List of sack implementations. Web Page of the Pittsburgh Supercomputing Center*.

http://www.psc.edu/networking/all_sack.html.

[2] Allcock, W., Bester, J., Breshahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., and Tuecke, S.
Secure, Efficient Data Transport and Replica Management for High-Performance Data_Intensive Computing. In *Proceedings of IEEE Mass Storage Conference*, 2001.

[3] *The AppLeS Homepage*
http://apples.ucsd.edu/

[4] *Automatic TCP Window Tuning and Applications. National Laboratory for Advanced Networking Research Web Page*
http://dast.nlanr.net/Projects/Autobuf_v1.0/autotcp.html

[5] Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Crummey, J., Reed, D., Torczon, L., and Wolski, R.
The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications, 15* (4). 327-344.

[6] Berman, F., Wolski, R., Figueira, S., Schopf, J., and Shao, G.
Application-Level Scheduling on Distributed Heterogeneous Networks. In *Proceedings of Proceedings of Supercomputing 1996*, 1996.

[7] Berman, F., and Wolski., R.
Scheduling from the Perspective of the Application. In *Proceedings of Proceedings of Symposium on High Performance Distributed Computing*, 1996.

[8] Boyd, E.L., Brett, G., Hobby, R., Jun, J., Shih, C., Vedantham, R., and Zekauska, M. *E2E piPEline: End-to-End Performance Initiative Performance Environment System Architecture.* July, 2002.
http://e2epi.internet2.edu/e2epipe11.shtml

[9] Casanova, H., Legrand, A., Zagorodnov, D., and Berman, F.
Heuristics for Scheduling Parameter Sweep applications in Grid environments. In *Proceedings of Proceedings of the 9th Heterogeneous Computing workshop (HCW'2000), pp349-363.*, 2000.

[10] Dail, H., Casanova, H., and Berman, F.
A Modular Scheduling Approach for Grid Application Development Environments. *Submitted to Journal of Parallel and Distributed Computing [In review]*.

[11] Dail, H., Casanova, H., and Berman., F.
A Decoupled Scheduling Approach for Grid Application Development Environments. In *Proceedings of Proceedings of Supercomputing, November 2002 [To appear]*.

[12] Dickens, P.
An Evaluation of Implementation Techniques for the FOBS Data Transfer Mechanism. In *Proceedings of.*

[13] Dickens, P.
A High Performance File Transfer Mechanism for Grid Computing. In *Proceedings of The 2002 Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA).* Las Vegas, Nevada, 2002.

[14] Dickens, P., and Gropp, B.
An Evaluation of Object-Based Data Transfers Across High Performance High Delay Networks. In *Proceedings of the 11th Conference on High Performance Distributed Computing*, Edinburgh, Scotland, 2002.

[15] Dickens, P., Gropp, B., and Woodward, P.
High Performance Wide Area Data Transfers Over High Performance Networks. In *Proceedings of The 2002 International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems.*, 2002.

[16] Dickens, P., and Thakur, R.
An Evaluation of Java's I/O Capabilities for High-Performance Computing. In *Proceedings of the ACM Java Grande 2000 Conference*.

[17] Dinda, P., and O'Hallaron, D. An Extensible Toolkit for Resource Prediction In Distributed Systems, Technical Report CMU-CS-99-138, Carnegie Mellon University, 1999.

[18] *Enabling High Performance Data Transfers on Hosts: (Notes for Users and System Administrators). Pittsburgh Supercomputing Center*
http://www.psc.edu/networking/perf_tune.html#intro

[19] Faerman, M., Su, A., Wolski, R., and Berman, F.
Adaptive Performance Prediction for Distributed Data-Intensive Applications. In *Proceedings of Proceedings of the ACM/IEEE SC99 Conference on High Performance Networking and Computing*, Portland, Or.

[20] Feng, W., and Tinnakornsrisuphap, P.
The Failure of TCP in High-Performance Computational Grids. In *Proceedings of Proceedings of Super Computing 2000 (SC2000)*.

[21] Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson, Ken Kennedy, Carl Kesselman, John Mellor-Crummey, Dan Reed, Linda Torczon, and Wolski., R.
The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications*, *15* (4). 327-344.

[22] *The Globus Project*
URL: http://www.globus.org

[23] *The Grid Applications Development Software Project Homepage*
http://juggler.ucsd.edu/~grads/

[24] Grimshaw, A., and Wulf, W.
Legion-a view from 50,000 feet. In *Proceedings of Proceedings of High-performance Distributed Computing Conference (HPDC)*, 1996.

[25] Hacker, T., Noble, B., and AAthey, B.
The Effects of Systemic Packet Loss on Aggregate TCP Flows. In *Proceedings of SC2002*, Baltimore Md., 2002.

[26] He, E., Leigh, J., Yu, O., and DeFanti, T.
Reliable Blast UDP : Predictable High Performance Bulk Data Transfer. In *Proceedings of Proceedings of the IEEE Cluster Computing*, Chicago, Illinois, September 2002.

[27] Holly Dail, Graziano Obertelli, Francine Berman, Rich Wolski, and Grimshaw, A.
Application-Aware Scheduling of a Magnetohydrodynamics Application in the Legion Metasystem. In *Proceedings of Proceedings of the 9th Heterogeneous Computing Workshop, May 2000.*, May, 2000.

[28] Jacobson, V., Braden, R., and Borman., D. TCP Extensions for high performance. RFC 1323, May 1992.

[29] Kennedy, K., Mazina, M., Mellor-Crummey, J., Cooper, K., Torczon, L., Berman, F., Chien, A., Dail, H., and Sievert., O.
Toward a framework for preparing and executing adaptive grid programs. In *Proceedings of Proceedings of NSF Next Generation Systems Program Workshop*, Fort Lauderdale, FL, 2002.

[30] Leigh, J., Yu, O., Schonfeld, D., and Ansari, R.
Adaptive Networking for Tele-Immersion. In *Proceedings of The Immersive Projection Technology/Eurographics Virtual Environments Workshop (IPT/EGVE)*, May, 2001.

[31] Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. TCP Selective Acknowledgement Options, RFC 2018. RFC 2018.

[32] *Modifying TCP's Congestion Control for High Speeds* May, 2002
http://www.aciri.org/floyd/

[33] Ostermann, S., Allman, M., and Kruse., H.
An Application-Level solution to TCP's Satellite Inefficiencies. In *Proceedings of Workshop on Satellite-based Information Services (WOSBIS)*, November, 1996.

[34] Postel, J. Transmission Control Protocol, RFC793.

[35] Sivakumar, H., Bailey, S., and Grossman, R.
PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. In *Proceedings of Super Computing 2000 (SC2000).*

[36] Sivakumar, H., Mazzucco, M., Zhang, Q., and Grossman, R.
Simple Available Bandwidth Utilization Library for High Speed Wide Area Networks. *Submitted to Journal of SuperComputing*.

[37] Su, A., Berman, F., Wolski, R., and Strout, M.M.
Using AppLeS to Schedule Simple SARA on the Computational Grid. *International Journal of High Performance Computing, 13* (3). 253-262. 1999

[38] Su, A., Berman, F., Wolski, R., and Strout., M.

Using AppLeS to Schedule Simple SARA on the Computational Grid. *International Journal of High Performance Computing Applications*, *13* (3). 253-262. 1999

[39] Vinkat, R., Dickens, P., and Gropp, B.
Efficient Communication Across the Internet in Wide-Area MPI. In *Proceedings of The 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA).* Las Vegas, Nevada, 2001.

[40] *The Web100 Homepage*
http://www.internet2.edu/e2epi/web02/p_web100.shtml