A Comparative Evaluation of High-Performance File Transfer Systems for Data-intensive Grid Applications

Cosimo Anglano, Massimo Canonico
Dipartimento di Informatica
Università del Piemonte Orientale, Alessandria (Italy)
email:{cosimo.anglano,massimo.canonico}@unipmn.it

Abstract

The ability of transferring very large files in the shortest amount of time is crucial for data-intensive Grid applications. Experimental evidence has shown that the mere availability of high-capacity wide-area networks is not sufficient to obtain adequate performance if vanilla TCP is used to transport data. Consequently, many alternative solutions are being explored, and a variety of data transfer tools have appeared. In this paper we experimentally compare some of these tools in various network scenarios. Our results show that solutions based on UDP, adopting rate-based algorithms, result in better performance than other alternatives in most cases, while solutions based on TCP are effective only under specific circumstances.

Keywords: data-intensive Grid applications, file transfer tools, high Bandwidth-Delay-Product Networks, TCP and UDP, PlanetLab

1 Introduction

The scientific exploration in many disciplines, like High Energy Physics, Climate Modeling, and Life Sciences, requires the processing of massive data collections. For instance, the amount of data that must be processed in many High-Energy Physics experiments is in the order of Terabytes (and sometimes even Petabytes) [5, 6, 7], while typical climate modeling applications generate output datasets whose size is in the order of hundreds of Gigabytes [10]. For these applications, the creation of *Data Grids*[12], that pool geographically distributed storage and computing resources, seems a promising solution. In order to enable the achievement of satisfactory performance, data-intensive grid applications require the availability of a system able to transfer potentially huge files in the shortest possible amount of time [10]. As a matter of fact, the completion time of typical data-intensive applications is given by the sum of their execution time and of the time taken to transfer the data they need [24], and is often dominated by the data transfer time.

An apparently straightforward way to reduce data transfer times is to increase the bandwidth of the networks interconnecting the resources in a Grid. Recent advances in communication technology are indeed enabling the deployment of Gigabit-per-second networks on geographic scale at a reasonable cost, and a few testbeds of this type have been already deployed. These networks, that are often called *high Bandwidth Delay Product (BDP) networks*, are characterized by very high round-trip latencies because of the distances covered by their links. However, empirical and analytical evidence exists demonstrating the intrinsic limitations of the TCP protocol, the de-facto standard for Internet transport, that has been shown to be unable to fully utilize the available capacity provided by high BDP networks. This is because TCP's windowing mechanisms imposes a limit on the amount of data it will send before it waits for an acknowledgment. Thus, the typical delays of high BDP networks imply that TCP will spend an inordinate amount of time waiting for acknowledgments, which in turn means that the client's data transmission will never reach the available capacity of the network. Therefore, simply increasing the available network bandwidth not necessarily translates into proportional improvements of data transfer performance, unless optimized TCP versions are used. The traditional solution to this problem is to adjust TCP's window and buffer sizes to match the BDP of the network, but not always this operation can be performed by users with standard privileges. For instance, in many Unix variants, the maximum buffer size is a system-wide value that can be modified only by the superuser. In order to overcome these intrinsic limitations of TCP, other approaches are being investigated, and new data transfer techniques are consequently being developed. These techniques work either at the system level or at the application-level. System level solutions, that usually require modifications to the operating system of the ma-

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

chines, of the network equipment, or of both, include automatic tuning of TCP window sizes [1], sometimes complemented by other sophisticated mechanisms [8], and development of new versions of TCP (e.g., Selective Acknowledgment TCP [23], High Speed TCP [16], and Scalable TCP [22]). This approach can yield very good performance, but usually requires significant updates of the networking infrastructure. Conversely, application level solutions do not require any modification to the networking infrastructure, but attempt to circumvent TCP problems by exploiting techniques not needing any special privilege. These techniques are based either on TCP or UDP. TCP-based solutions (e.g., GridFTP [26], bbFTP [4], and bbcp [19]) use TCP connections to move data, and attempt to circumvent TCP's window size problems by using parallel streams, so that an aggregate congestion window, matching the BDP on the network path used to carry out the transfer, is obtained. Conversely, UDP-based solutions (e.g., FOBS [14], Tsunami [27], UDT [17], and SABUL [18], use UDP to move data, and employ rate-based control algorithms to match the transmission speed with the available network bandwidth, sometimes coupled with congestion and flow control algorithms in order to keep as low as possible the loss rate during transmissions. for congestion control.

Given the relatively large number of high-performance transfer tools, the question about the effectiveness of each of them arises naturally. However, although prototypes of many of the above systems have been around for a while, an experimental comparison is still lacking in the literature. To the best of our knowledge, the only study of this type has been carried on Lambda-Grids [21], whose peculiarities make the corresponding results inapplicable to more conventional Grid networking infrastructures. This paper aims at filling this gap by presenting the results we collected by performing data transfer experiments, among machines of the PlanetLab [13] testbed, using some of the above tools that are, in our opinion, representative of their class. In our investigations we tried to answer the following questions:

1. Are TCP-based and UDP-based solutions equivalent, or one approach yields better performance than the other one, when production networks are used?

2. Is there a data transfer tool that works better than the other ones?

3. How do these tools work for production networks with relatively low BDP values?

In the rest of this paper we will provide preliminary answers to the above questions, using the results we collected. Our results indicate that, while for relatively low BDP networks both TCP-based and UDP-based approaches achieve similar performance (although the latter ones are more effective), for higher BDP networks UDP-based tools are definitely the only viable option.

The rest of the paper is organized as follows. In Section 2 we briefly describe the main characteristics of the tools we compared. In Section 3 we discuss the experimental methodology we used, report the results we got, and highlight the differences among the various tools. Finally, Section 4 concludes the paper and outlines future research work.

2 File transfer tools

As anticipated in the Introduction, our study focuses on application-level data transfer systems that do not require kernel modifications. To the best of our knowledge, the exhaustive list of these systems is: FOBS [14], Tsunami [27], SABUL [18], UDT [17], RBUDP [20], GTP [28], GridFTP [26], bbcp [19], bbFTP [4], and PSockets [25]. In our study, however, we considered only a subset of the above tools, namely FOBS, UDT, RBUDP, and bbFTP, for the following reasons. Tsunami seems to be a very promising tool, but its current implementation suffers from a bug [11] that makes it hang (and slow down also other processes) from time to time, making it hard to perform an extensive experimentation. UDT is the successor of SABUL, so it should exhibit higher performance, while GTP is specific for Lambda Grids and focuses on multipoint-to-point transfers. Among the TCP-based tools, we selected bbFTP as representative of the whole class, since we believe that all of them offer similar performance. Furthermore, GridFTP requires the installation of various Globus components, we did not have access to any PSockets implementation, and bbcp posed us some problems in using it through scripts, so these tools were not considered in our evaluation. In the rest of this Section, we will describe in more detail the tools we compared.

2.1 bbFTP

bbFTP is a file transfer system, developed by the High-Energy Physics community, that uses multiple TCP connections between the same endpoints to speed-up data transfers. It uses several aggressive optimization techniques (like on-the-fly data compression and automatic sizing of TCP windows) to improve performance, and provides a set of advanced functionalities concerning authentication and interfacing with various I/O subsystems. We tested version 3.0.2, downloaded on Jan. 27th, 2004.

2.2 Reliable Blast UDP (RBUDP)

The Reliable Blast UDP system, a part of the Quality of Service Adaptive Networking Toolkit (QUANTA) [3], be-

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

longs to the family of UDP-based solution. With RBDUP, hosts exchange data packets via UDP, and control packets via TCP. More specifically, a transfer is carried out in the following way. In the first data transmission phase, the source machine sends the whole file, at a constant, user-specified rate, to the machine requesting it. At the end of this phase, the receiver sends back (via a TCP connection) a list of lost packets, that are retransmitted again as a "blast". This process is repeated until no more packets need to be retransmitted. RBDUP requires the user to specify the rate at which data are sent, and this rate is kept constant for the whole transfer duration. RBUDP authors recommend to specify a sending rate not larger than the bandwidth of the bottleneck link of the path connecting the two hosts, that must be determined prior to the transfer by using a suitable tool like Iperf [2] (or its extended version app_perf developed as part of RBUDP). We tested the version distributed with QUANTA 0.3, downloaded on Feb. 18th, 2004.

2.3 FOBS

FOBS is a user-level, UDP-based data transfer mechanisms in which, unlike RBUDP, there is some degree of interaction between the sender and the receiver during the transfer. More specifically, the sender works by transmitting (as fast as possible) a batch of fixed size data packets, that includes both unsent and retransmitted packets. Periodically, the receiver sends acknowledgment packets to the sender, containing a list of unreceived packets, that is used by the sender to decide how many lost packets will be resent in the the next batch. The version of FOBS used in our experiments includes also a congestion control mechanism [15] used to adjust the sender's speed to the conditions of the network. We tested the version downloaded on Jan. 12th, 2004.

2.4 UDP-based Data Transfer protocol (UDT)

The UDT protocol, the latest version of the Simple Available Bandwidth Utilization Library (SABUL) [18], uses UDP to transfer data between pairs of hosts, and combines rate-based, window-based and delay-based control mechanisms to deliver high throughput and low loss data transmission. UDT implements slow start and AIMD control schemes for flow control (which makes it more TCP friendly than the other rate-based schemes), and window-based control for controlling the number of outstanding packets in flight. UDT employs also rate adjustment based on delay monitoring, providing improved performance over classical AIMD schemes. More specifically, the sender calculates the inter-packet time, which is updated by the rate-control algorithm. Packets are sent out every inter-packet time, but the number of outstanding packets cannot exceed the threshold imposed by the window size, that in turn is dynamically adjusted to match the current network conditions. The receiver reorders, if necessary, the packets it receives, and uses selective acknowledgments, sent at a constant rate, to request lost packets. We tested version 1.1, downloaded on Jan. 12th, 2004.

3 Experimental evaluation

As anticipated in the Introduction, the goal of our study was to compare the above tools from the perspective of a user that has the need of transferring large files over production networks. Therefore, in our evaluation we considered only user-perceived performance indices, and in particular the throughput achieved during the transfer, and not other figures of merit that could give insights into the behavior of each tool (e.g., fairness w.r.t. other traffic flows, fast adjustment to network capacity, etc.).

3.1 Experimental setup

Our performance comparison has been carried out by deploying the above tools on a set of machines belonging to the PlanetLab testbed, and by running a set of transfer experiments, spanning about two weeks, aimed at measuring the throughput achieved by each tool.

In order to compare these tools in a variety of scenarios, we used both machines connected by paths with a relatively high BDP value (around 3.75 MBytes), and machines connected by paths with lower BDP values (from 250 to 800KBytes). The characteristics of the testbed we used for our experiments are reported in Table 1, where rows correspond to "clients" (i.e., machines requesting the transfer of files), columns correspond to "servers" (i.e., machines from which the transfers start upon request from a client). For each (*client,server*) pair, we report both the Round-Trip-Time (RTT) and path capacity (C), that have been measured by means of the pathrate [9] tool. The BDP for each pair is computed as the product of the RTT and the path capacity. Note that in Table 1 we use short names for the involved machines, in order to keep the table within the page margins. The correspondence between short names and full names is reported in Table 2.

All the machines used in the experiments had identical configurations (they all run the PlanetLab software). In particular, the maximum read/write buffer size for TCP sockets was set to 128KBytes. The only (notable) difference among these machines is represented by the Taiwan host, whose incoming and outgoing bandwidth is limited to 10 Mbit/sec., although this has not been detected by pathrate that estimated a path capacity of about 100 Mbit/sec.

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

	Taiwan		Atlanta		Houston		Montreal	
	RTT	C	RTT	C	RTT	C	RTT	C
Ottawa	300	97	49	98	65	98	23	101
Atlanta	230	99	–	–	20	100	49	100
Taiwan	–	–	229	97	233	95	243	96

Table 1. Performance characteristics of the testbed used for experiments. Round-trip-times (RTT) are expressed in milliseconds, path capacities (C) in Mbit/sec.

Short name	Full address
Montreal	planet1.montreal.canet4.nodes.planet-lab.org
Ottawa	planet1.ottawa.canet4.nodes.planet-lab.org
Atlanta	planetlab1.atla.internet2.planet-lab.org
Houston	planetlab1.hstn.internet2.planet-lab.org
Taiwan	planetlab1.im.ntu.edu.tw

Table 2. Correspondence among short and full names of machines used for the experiments

3.2 Evaluation methodology

Our evaluation has been carried out by downloading from each "client" machine a 100 MBytes file using each of the tools we considered. Since the performance of tools using parallel TCP streams strongly depend on the number of TCP connections used to carry out the transfer, for bbFTP we ran experiments using 5, 10 and 15 parallel streams. Thus, from each client we performed six different download experiments.

Performing a comparison based on measurements collected on production networks is not an easy task, since the presence and the variation of cross traffic may differently affect the various tools, thus biasing the results. In order to minimize these effects, we organized our experiments in *rounds*. In a given round, we chose a client and performed "back-to-back" all the six download experiments. Moreover, the relatively small size of the files used for the experiments (100 MBytes) allowed us to keep the duration of each download experiment within a few minutes in most cases, so that an entire round could be completed in a relatively short time. In this way, we tried to minimize the probability of dramatic cross-traffic changes between the downloads performed with different tools. We performed a sequence of rounds, spanning about two weeks, during which we collected the throughput achieved during the various transfers, and used these measurements to compute average values. In order to quantify the possible effects of cross traffic flows, we computed also the standard deviations of the measured values. Intuitively, higher standard deviations indicate larger variations of cross traffic, while smaller values indicate a relative stability of network conditions across the various round we performed.

3.3 Experimental results

The results obtained for our experiments are reported in Table 3, where each row corresponds to a particular (*client,server*) pair and reports the average throughput value, and the respective standard deviation, for each of the tools we considered. For each row, the tool that achieved the best performance is highlighed by using bold fonts for the corresponding throughput value. As shown by the results in the table, the best tool for paths characterized by a relatively small BDP value (from 250 to 800 KBytes), i.e. all the paths that did not include Taiwan as endpoint, was either RBUDP or UDT. More specifically, RBUDP provided better results when the client was Atlanta (its throughput was from 2% to 6% higher than those obtained by UDT), while UDT achieved better performance when the client was Ottawa (its throughput was from 8.9% to 16.2% higher). The performance of bbFTP for these paths can be considered quite good as well, although the lack of a monotonic trend in the throughput values obtained for increasing numbers of parallel streams reveals the intrinsic drawback of this approach, namely the difficulty in choosing an appropriate value for this parameter. For instance, the best performance for the Atlanta client, downloading either from Ottawa or from Montreal with bbFTP, were obtained with 10 parallel streams (bbFTP-10 column), while for the download from Houston 5 streams were sufficient. Finally, the performance of FOBS on lower BDP paths was quite disappointing, as in these cases it achieved a throughput 8 or 9 times smaller than UDT, RBUDP, or bbFTP, showing that UDP-based solutions that do not adopt a rate-based approach in order to match the transmission speed to the available network bandwidth, as instead done by RBUDP and UDT, may fail to achieve adequate performance. The small values for the standard deviations computed for these paths indicate also that the results were not polluted by cross-traffic variations.

For paths with larger BDP values (around 3.75 MBytes), i.e. all the paths including Taiwan as endpoint, the situation becomes a bit different. In particular, when Taiwan acted as server, RBUDP and UDT showed very similar performance, while the other tools obtained a smaller throughput, but were more or less equivalent to each other. However, when Taiwan was the client, the differences among the various tools became very evident. In particular, the best tool was by far RBDUP that, with 25 Mbit/s., showed improvements from 124% to 137% w.r.t. UDT, from 219% to 394%

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

Hosts		FOBS		RBUDP		UDT		bbFTP-5		bbFTP-10		bbFTP-15	
Client	Server	Avg	St.Dev.	Avg	St.Dev.	Avg	St.Dev.	Avg	St.Dev.	Avg	St.Dev.	Avg	St.Dev.
Atlanta	Taiwan	7.17	0.51	**10.34**	**1.82**	10.11	1.43	6.36	1.62	7.97	1	9.62	0.56
Atlanta	Ottawa	8.82	0.81	**86.25**	**3.63**	81.03	3.06	79.48	2.97	79.70	3.29	76.98	3.15
Atlanta	Montreal	9.66	0.2	**86.71**	**1.64**	82.31	2.86	72.59	0.52	86.04	0.14	72.56	4.08
Atlanta	Houston	9.34	0.47	**86.29**	**2.47**	84.28	3.3	87.94	0.51	85.62	1.19	81.35	1.88
Ottawa	Taiwan	7.24	0.42	**10.58**	**0.98**	9.98	1.64	4.98	1.36	8.23	1.03	9.34	0.92
Ottawa	Atlanta	9.17	0.43	75.87	19.24	**86.02**	**3.17**	79.63	5.4	77.78	4.37	80.37	5.15
Ottawa	Montreal	9.71	0.22	77.74	19.77	**90.4**	**5.59**	88.11	3.66	88.99	1.79	89.4	1.76
Ottawa	Houston	9.27	0.4	76.15	19.13	**82.96**	**4.49**	81.35	5.11	76.43	5.69	73.85	3.66
Taiwan	Ottawa	5.7	2.8	**25.06**	**8.78**	11.18	2.1	< 1	–	1.26	0.13	1.63	0.29
Taiwan	Atlanta	7.1	2.56	**22.36**	**9.28**	11.17	1.85	< 1	–	1.3	0.14	1.78	0.22
Taiwan	Montreal	6.48	2.98	**25.29**	**6.93**	10.65	2.98	<1	–	1.28	0.14	1.72	0.2
Taiwan	Houston	7.18	2.48	**21.73**	**8.71**	10.06	1.87	<1	–	1.25	0.15	1.65	0.26

Table 3. Experimental results. Average throughputs are expressed in Mbit/s.

w.r.t. FOBS, and from 1156% to 1437% w.r.t. bbFTP. This behavior might be due to the effects of the mechanisms used by PlanetLab nodes to enforce bandwidth limitations. This hypothesis seems to be supported by the fact that, not considering RBUDP, the throughput achieved by all the tools does not exceed the bandwidth limit of 10 Mbit/sec. imposed on the Taiwan machine. These mechanisms seem to be effective only for those tools that employ some form of congestion control (in practice, all the tools but RBUDP), while they appear ineffective for RBUDP that, as already discussed, does not encompass any congestion control feature and hence might be able to unhinge them. As additional evidence to this hypothesis, consider that for all the tools the observed standard deviation was quite low, indicating the stability of the network conditions across all the experiments, while for RBDUP was considerably higher, possibly indicating that sometimes the limiting mechanisms had some effects also on RBUDP, but that in most cases they were ineffective. However, additional experiments are required in order to better understand this behavior.

In summary, our results indicate that, while for relatively low BDP networks both TCP-based and UDP-based approaches achieve similar performance (although the latter ones are more effective), for higher BDP networks UDP-based tools are definitely the only viable option. However, simply using UDP does not automatically result in good performance, as demonstrated by FOBS that, not using any rate-based mechanism, was not able to achieve performance comparable with those attained by the other UDP-based tools. As a final consideration, we note that using UDT is definitely simpler than using RBUDP, as it does not require the user to perform any estimation of the available bandwidth on the network path connecting the hosts involved in the transfer.

4 Conclusions and future work

In this paper we presented a comparison of available high-performance data transfer tools for data-intensive Grid applications. Although these tools have been developed for very high BDP networks, and hence when used on relatively low BDP networks may show unexpected and unplanned behaviors, we believe that our comparison makes sense from a practical point of view. We indeed believe that the vast majority of Grid users will not have access to very high-speed networking infrastructures, so a comparison of available high-performance transfer tools on production networks may provide them with valuable information.

The results presented in this paper, however, must be considered preliminary. A more thorough experimentation, aimed at better characterizing the behavior of the various tools and at comparing their performance, is indeed planned for the near future. In this experimentation, we plan to use larger files (from 500MBytes to a few GBytes). However, this requires the development of suitable measurement techniques able to factor out the possible effects of cross-traffic variations that almost certainly will arise for long-lasting transfers. Moreover, we plan to include more machines in our testbed, in order to be able to draw more general conclusions about the performance of the various tools. Finally, we plan to perform data transfer experiments on network paths exhibiting very large BDP values, much larger than those currently characterizing PlanetLab, although finding and accessing these testbeds appears, at the moment, challenging.

References

[1] Automatic TCP Window Tuning and Applications. http://dast.nlanr.net/Projects/Autobuf_v.10/autotcp.html.

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

[2] Iperf: The TCP/UDP Bandwidth Measurement Tool. http://dast.nlanr.net/Projects/Iperf.

[3] QUANTA Home Page. http://www.evl.uiuc.edu/cavern/quanta. Accessed on March 28th, 2004.

[4] The bbFTP – Large Files Transfer Protocols Web Site. http://doc.in2p3.fr/bbftp.

[5] The Grid Physics Networks (GriPhyN) project. http://www.griphyn.org.

[6] The Large Hadron Collider (LHC) project. http://lhc.web.cern.ch/lhc.

[7] The Particle Physics Data Grid Project. http://www.cacr.caltech/edu/ppdg.

[8] The Web100 Project. http://www.web100.org/.

[9] The BW-meter project Home Page. http://www.pathrate.org, 2004.

[10] 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 Computational Grid Environments. *Parallel Computing*, 28(5):749–771, May 2002.

[11] S. Ansari. Tsunamy - A Study. http://www-iepm.slac.stanford.edu/bw/Tsunami.htm.

[12] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23:187–200, 2001.

[13] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communications Review*, 33(3), July 2003. http://www.planet-lab.org.

[14] P.M. Dickens. FOBS: A Lightweight Communication Protocol for Grid Computing. In *Proc. of Europar 2003*, Klagenfurt, Austria, August 2003.

[15] P.M. Dickens and V. Kannan. Application-Level Congestion Control Mechanisms for Large Scale Data Transfers Across Computational Grids. http://www.cs.iit.edu/ pmd/FOBS/fobs_papers.html, 2004. Submitted for publications.

[16] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Dec. 2003.

[17] Y. Gu and R.L. Grossman. Using UDP for Reliable Data Transfer over High Bandwidth-Delay Product Networks. http://www.dataspaceweb.net/papers.htm, 2003. Submitted for publication.

[18] Y. Gu, X. Hong, M. Mazzucco, and R.L. Grossman. SABUL: A High Performance Data Transfer Protocol. http://www.dataspaceweb.net/papers.htm, 2003. Submitted for publication.

[19] A. Hanushevsky, A. Trunov, and L. Cottrell. Peer-to-Peer Computing for Secure High Performance Data Copying. In *Proc. of the 2001 Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP 2001)*, Beijng, China, September 2001.

[20] E. He, J. Leigh, O. Yu, and T. DeFanti. Reliable Blast UDP: Predictable High Performance Bulk Data Transfer. In *Proc. of 5th Int. Conf. on Cluster Computing*, 2002.

[21] R. Huang and A. Chien. Benchmarking High Bandwidth Delay Product Networks. Technical report, Concurrent Systems Architecture Group, University of California, San Diego. Retrieved on March 28th, 2004.

[22] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. In *Proc. of First Int. Workshop on Protocols for Fast Long-Distance Networks*, CERN, Geneva, Switzerland, February 2003.

[23] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment. RFC 2018.

[24] K. Ranganathan and I. Foster. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. *Journal of Grid Computing*, 1(1):53–62, 2003.

[25] 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. In *Proc. of Supercomputing 2000*, 2000.

[26] The Globus Team. GridFTP: Universal Data Transfer for the Grid. http://www.globus.org. White Paper.

[27] S. Wallace. Tsunami File Transfer Protocol. In *Proc. of First Int. Workshop on Protocols for Fast Long-Distance Networks*, CERN, Geneva, Switzerland, February 2003.

[28] R.X. Wu and A. Chien. GTP: Group Transport Protocol for Lambda-Grids. In *Proc. of 4th ACM/IEEE Int. Symp. on Cluster Computing and the Grid*, Chicago,USA, 2004.

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE