IN-62-CR
43729

RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Achieving High Throughput for Data Transfer over ATM Networks

Marjory J. Johnson
Jeffrey N. Townsend

RIACS Technical Report 96.08

March 1996

Accepted for ICC '96, June 1996

Achieving High Throughput for Data Transfer over ATM Networks

Marjory J. Johnson
Jeffrey N. Townsend

The Research Institute for Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 212, Columbia, MD 21044, (410)730-2656

Work reported herein was supported by the National Aeronautics and Space Administration under Contract NAS 2-13721 to the Universities Space Research Association (USRA).

Achieving High Throughput For Data Transfer Over ATM Networks

*Marjory J. Johnson and Jeffrey N. Townsend**
RIACS
NASA Ames Research Center
Moffett Field, CA 94035 -1000
U.S.A.

Abstract. File-transfer rates for ftp are often reported to be relatively slow, compared to the raw bandwidth available in emerging gigabit networks. While a major bottleneck is disk I/O, protocol issues impact performance as well. Ftp was developed and optimized for use over the TCP/IP protocol stack of the Internet. However, TCP has been shown to run inefficiently over ATM. In an effort to maximize network throughput, data-transfer protocols can be developed to run over UDP or directly over IP, rather than over TCP. If error-free transmission is required, techniques for achieving reliable transmission can be included as part of the transfer protocol. However, selected image-processing applications can tolerate a low level of errors in images that are transmitted over a network. In this paper we report on experimental work to develop a high-throughput protocol for unreliable data transfer over ATM networks. We attempt to maximize throughput by keeping the communications pipe full, but still keep packet loss under five percent. We use the Bay Area Gigabit Network Testbed as our experimental platform.

1. Introduction

As people experimenting with gigabit testbeds soon discover, even though network bandwidth is 155 Mbps or greater, it is almost impossible to achieve the maximal theoretical throughput for an end-to-end application. While some impressive throughput figures are reported from such benchmark tools as ttcp or netperf [8], (e.g., see [15] for BAGNet performance statistics), application throughput is often significantly lower. Even a simple application like ftp often exhibits relatively low throughput. Examples of potential performance bottlenecks include disk I/O, workstation compute power, and network-protocol limitations. We focus on network-protocol issues in this paper.

Network protocols that have been fine tuned for use over the Internet need not work well over ATM networks. For example, a frequently cited simulation study by Romanow and Floyd [11] predicts potential throughput for TCP/IP over ATM as low as 34%. The reason for this is the basic mismatch between the unit of transmission for the ATM network, a 53-byte cell, and the much larger TCP/IP packet. If even a single cell of a TCP/IP packet is lost, the entire packet will be retransmitted. This would increase the network congestion that probably caused the cell loss to begin with. As another example, Moldeklev and Gunningberg [10] report that heuristic algorithms that have been developed to enhance performance on the Internet may even create a deadlock situation on ATM networks.

At RIACS we are exploring the use of high-speed ATM networks to support NASA scientific endeavors. We are currently investigating the use of networks to support the joint interactive analysis of large image files by Earth scientists located at different geographical sites. One element of such collaborative work is the transfer of an image file from one location to another, or perhaps the downloading of an image file to multiple sites from a remote database. It is this data-transfer aspect that we address in this paper. Since transfer of the image file is part of an interactive session, achieving high throughput is important. However, unlike many file-transfer applications, error-free transfer is not necessarily critical. For example, since image classification is an inexact process, results from a classification algorithm are unlikely to be significantly altered if a few packets are lost during transfer of the image over a network.

The study reported herein is an experimental performance analysis of data transfer over ATM networks. The paper is organized as follows. In Section 2 we describe our experimental platform, the Bay Area Gigabit Network Testbed. In Section 3 we present a simple buffer-based protocol for file transfer. In the context of this protocol, we determine how the sender's

* This work was supported by the National Aeronautics and Space Administration (NASA) under Contract NAS2-13721 between NASA and the Universities Space Research Association (USRA).

time is apportioned among the basic components of the transfer process. In Section 4, we describe an unreliable data-transfer protocol that is designed to achieve maximal throughput, while keeping packet loss manageable, by keeping the communications channel full. We compare the resulting throughput with throughput using ftp. Finally, in Section 5, we compare our approach to achieving high throughput for data transfer with previous work reported in the literature. We conclude by outlining future plans for further protocol development.

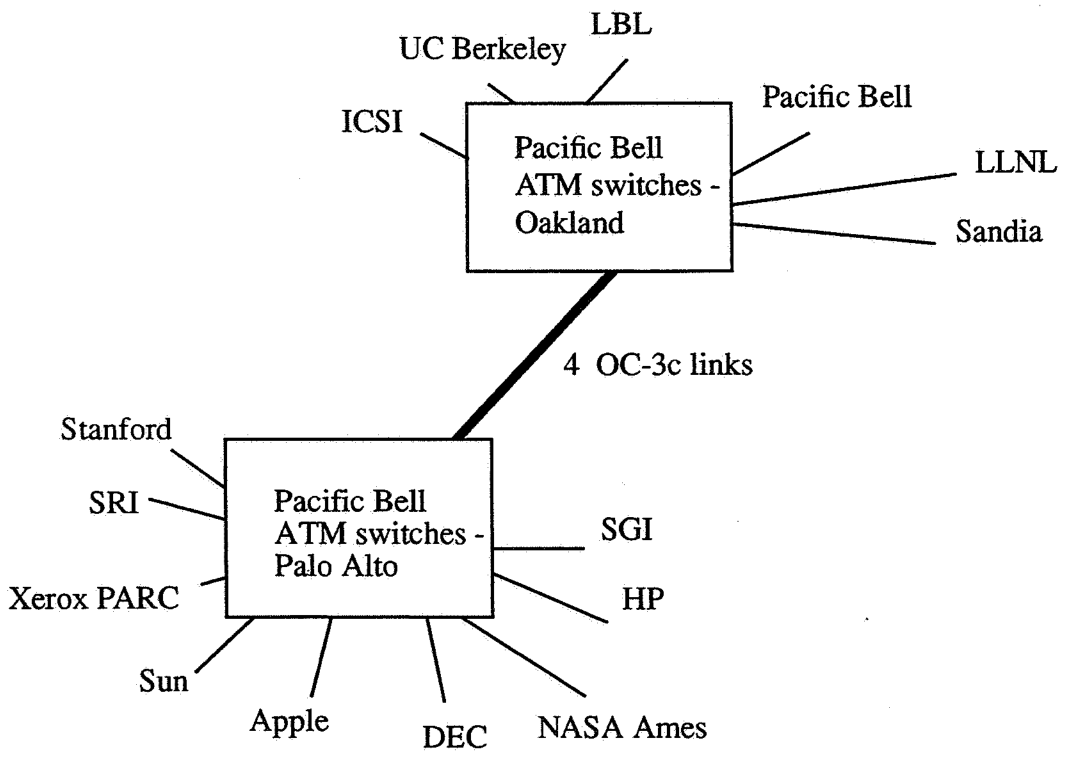
2. Experimental Platform

We are using the Bay Area Gigabit Network Testbed (BAGNet) as our experimental platform. BAGNet is a high-performance ATM (155 Mbps) testbed located within the San Francisco Bay Area in northern California. BAGNet is a metropolitan-area network, with up to approximately 50 miles separating pairs of participating sites. It is sponsored by Pacific Bell's California Research and Education Network (CalREN) program. Under this program Pacific Bell is providing the basic infrastructure for several ATM testbeds within the state of California, to promote the development of high-performance communication applications. BAGNet is a metropolitan-area network, including sites in both the South and East Bay areas. Pacific Bell is providing a pair of ATM switches in the South Bay and a pair of ATM switches in the East Bay, connected by a double mesh of OC-3c links. The switches provided by Pacific Bell are Newbridge switches. An individual BAGNet site typically has a small number of hosts connected to a local ATM switch, which is then directly connected to one of the Pacific Bell switches via a SONET OC-3c (155 Mbps) link. Figure 1 illustrates the overall testbed configuration.

Performance of applications running on BAGNet is, of course, dependent on the testbed's protocol infrastructure. BAGNet is an IP over ATM testbed, with AAL5 as the adaptation layer and LLC/SNAP encapsulation, in accordance with RFCs 1483 and 1577. We are using only Permanent Virtual Channels (PVCs) for transmission over BAGNet. A mesh of PVCs enables every host on BAGNet to have a permanent connection to every other testbed host. Hence, our performance numbers do not include any VC switching overhead.

In addition, we are not imposing any congestion control on any of the point-to-point BAGNet traffic. If

Figure 1. BAGNet Testbed Configuration



the network becomes congested, the ATM switches will simply drop cells. A few months ago Pacific Bell was using bandwidth policing on the individual PVCs, limiting the amount of buffer space allotted to each PVC. Since this practice caused serious performance degradation, as documented on the World Wide Web (WWW) [2], the policing has been removed. For more information about BAGNet, see the BAGNet home page [1], various other WWW pages that are linked to [1], or Johnson [7].

With a testbed such as BAGNet, there are many variables that affect end-to-end performance, including workstation hardware and software, testbed interfaces, number of ATM switches in the end-to-end communications path, etc. To evaluate the impact of some of these variables, we conducted all our experiments using a variety of workstations at three different testbed sites. We used a Sun Sparc 2 and a Sun Sparc 10 at NASA Ames Research Center, a Sun Sparc 10 at Sandia National Laboratories - Livermore, and a Sun Sparc 20 at Lawrence Livermore National Laboratory (LLNL). Table 1 summarizes pertinent information about each of these workstations and their interfaces to BAGNet.

3. Basic Components of Data Transfer

In this section we describe a simple unreliable data-transfer protocol that provides a low level of synchronization to minimize transmission errors. Using this protocol, we measure the percentage of time that can be attributed to each of the major components of the data-transfer process.

First we determine the level of performance that is achievable when using raw UDP to transfer 100 Megabytes of data. We graphed throughput versus packet size for all possible sender/receiver combinations. Transmission errors, causing the throughput to be lower at the receiver than at the sender, are especially evident when there is a mismatch in either workstation capabilities or the type of network interface. While the actual throughput values varied, depending on the particular workstations, the basic shapes of the plots are similar. Figures 2 - 4 are representative of our results. Maximal throughput is achieved for packet size of approximately 9000 bytes, as expected, given that the default MTU for IP over ATM is 9180 bytes [9].

When the sending and receiving workstations are the same type, as in Figure 2, where both workstations are Sun Sparc 10s, sender and receiver throughput are almost identical. In this case, the percentage of packet errors was consistently below 5%. We observed the same result if the receiving workstation is the more powerful of the pair. However, if the receiving workstation is less powerful, significant packet loss can occur, as illustrated in Figure 3. The percentage of packet errors in this case is clearly unacceptable. Figure 4 illustrates the fact that the Sun Sparc 20 with the OC-3c (155 Mbps) interface will completely overrun a TAXI (100 Mbps) interface. The ATM switches simply do not contain enough buffers to cope with the mismatch in line speeds. Figures 3 and 4 clearly show that if the more powerful workstation is the sender, some kind of flow control must be used in order to prevent the faster workstation from overrunning the slower one.

Any protocol to transfer a file must necessarily include disk I/O; accordingly, it is unrealistic to expect to be able to match raw UDP throughput for file transfer. However, the above numbers do provide a goal to aim for. Since we wish to minimize the rate of errors, without imposing the overhead that would be required to ensure completely reliable data transfer, we developed a protocol that periodically synchronizes the sender and receiver to prevent the sender from overrunning the receiver.

Table 1: Workstation Characteristics

Workstn Name	Site	Workstn Type	Operating System	Site Switch	Network Interface
psyche	NASA	Sparc2	SunOS 4.1.4	Fore ASX200	Fore/TAXI
dufus	NASA	Sparc10	SunOS 4.1.3	Fore ASX200	Fore/TAXI
conan	Sandia	Sparc10	SunOS 4.1.3	Fore ASX200	Fore/TAXI
chips	LLNL	Sparc20	Solaris 2.3	Fore ASX200	Fore/OC-3c

Figure 2. UDP Throughput: dufus to conan

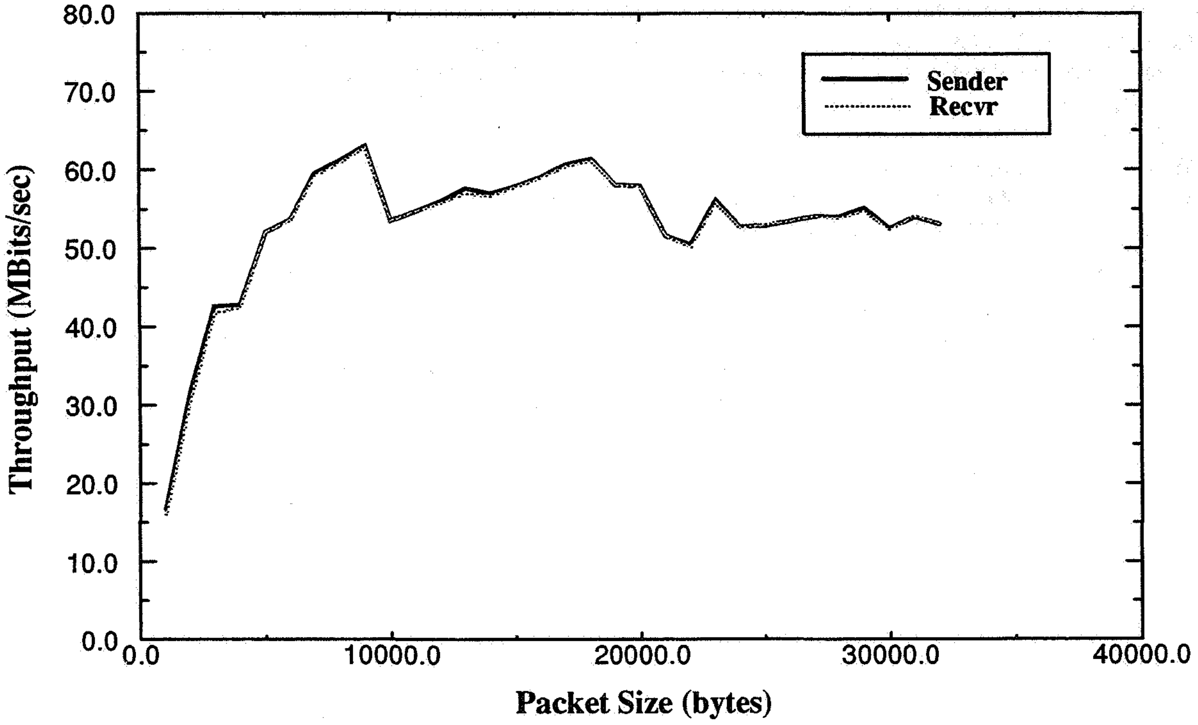


Figure 3. UDP Throughput: dufus to psyche

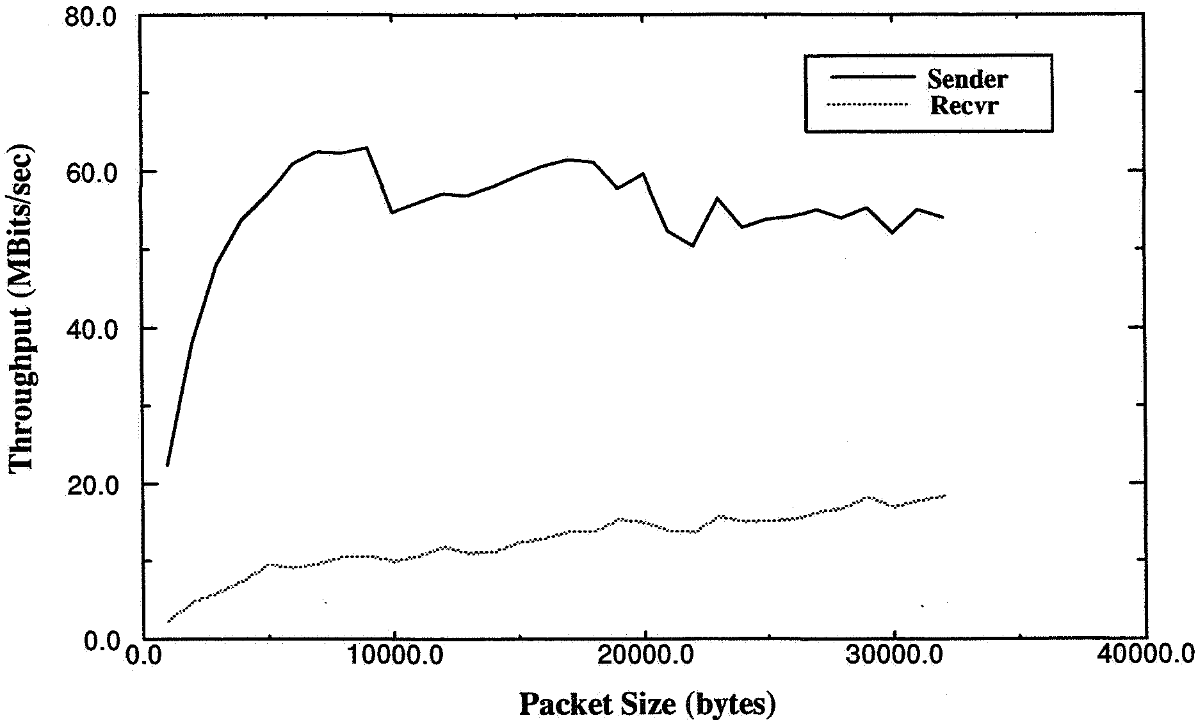
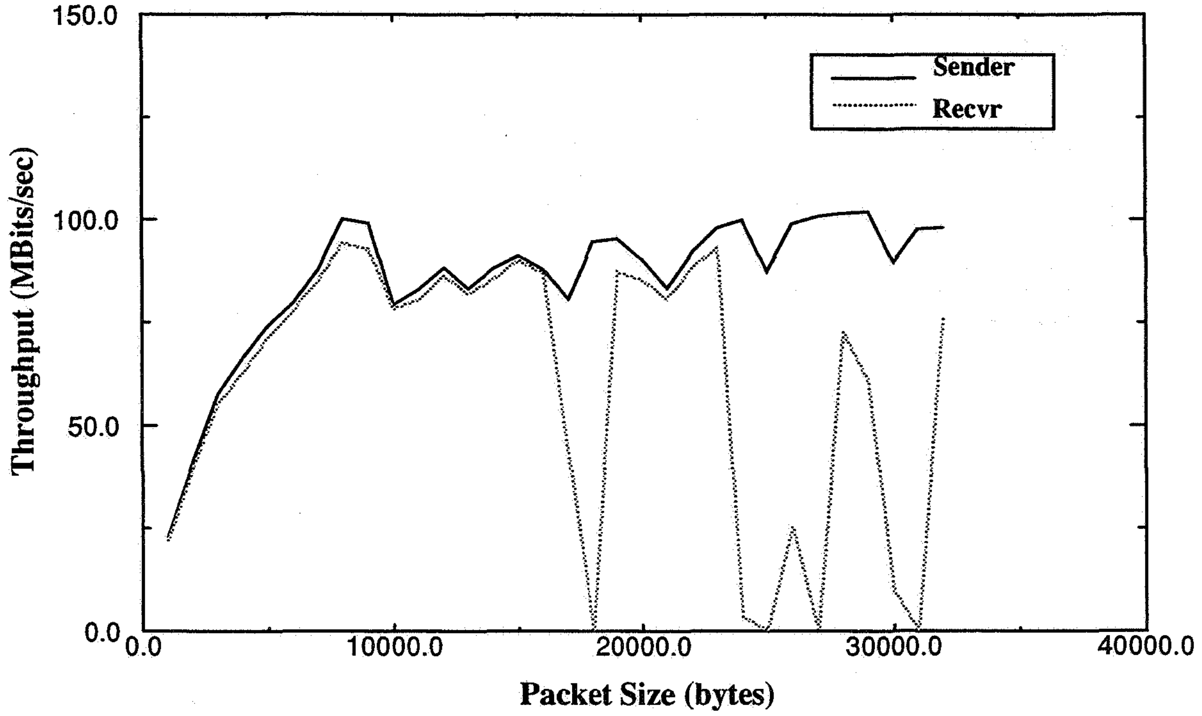


Figure 4. UDP Throughput: chips to conan



Our Simple Buffer Transfer Protocol (SBTP) is based on the transfer of large buffers, similar to NETBLT [4]. These buffers are broken down into 9000-byte packets for IP transfer. However, whereas NETBLT is a reliable protocol, providing for retransmission of lost packets at the end of each buffer transfer, our protocol is unreliable. The sending and receiving workstations simply synchronize their activities at the beginning and end of each buffer, as a means of minimal flow control. This synchronization is done reliably, using a TCP control socket. At the beginning of a buffer, the sender uses TCP to inform the receiver of the buffer size and the packet size. When the sender receives a ready signal from the receiver, it uses UDP to transmit all the data in the buffer in packet-sized segments and then signals the end of the buffer. Upon receipt of the end-of-buffer signal, the receiver writes the buffer to the disk and then waits for another start-of-buffer signal from the sender.

To further control packet loss from buffer overflow during transmission, we striped the data from each send buffer over multiple sockets at the receiver. Our experiments show that as the number of sockets is increased, the percentage of lost packets decreases until it is approximately one to two percent, while the throughput remains virtually constant.

Since writing to disk is normally slower than reading from disk, we would expect that the sender would have to wait for the receiver in between sending of consecutive buffers. We used SBTP to transfer a 100 Megabyte file between all possible pairs of workstations listed in Table 1. In addition to total throughput, we are interested in the percentage of total elapsed time that the sender devotes to each of the basic components of the protocol. Table 2 presents a representative subset of our results. For each source/destination pair, the sum of the percentages is less than 100%, because we ignore the time occupied by miscellaneous minor activities.

The table entries in the column labeled "sync" give the portion of time spent synchronizing the sender and receiver. The timer is set after the sender indicates to the receiver that it is ready to send. The timer is reset after the receiver indicates that it is ready to receive. "Disk I/O" time is the portion of time spent transferring data from the disk to the send buffer. "Data transfer" time is the percentage of time that the sender is actively transferring data over the network, including the time required for protocol processing and processing in the kernel, as well as the time that the data is physically present on the network. Note that this is the only portion of the sender's time that utilizes network resources. Because data striping results in minimal packet loss, sender and receiver throughput are virtually the same. Thus, only a single throughput value is recorded in the table.

Table 2: Basic Components of Transfer Process

Source/Dest	Sync (%)	Disk I/O (%)	Data transfer (%)	Total throughput
psyche/conan	13.4%	27.8%	55.0%	12.4 Mbps
psyche/dufus	18.9%	25.0%	43.0%	10.8 Mbps
dufus/conan	19.38%	30.7%	47.0%	11.0 Mbps
conan/dufus	29.6%	18.5%	49.3%	14.9 Mbps

Table 2 shows that a large portion of the time required for file transfer is actually dedicated to peripheral activities. In all our experiments approximately half the sender's time was spent doing nonproductive work, i.e., doing tasks that don't utilize network resources. This seems to indicate that maximum achievable throughput is approximately double that reported in the table. However, this is really only a lower bound for possible improvement, since the "data transfer" portion of time could potentially be reduced by increasing the efficiency of the data-transfer algorithm, streamlining the workstation/network interface, using a different operating system, etc.

4. An Unreliable Data Transfer Protocol and its Performance

The fact that the network is idle such a large percentage of the time when using the Simple Buffer Transfer Protocol motivates the development of the final protocol reported in this paper. The Multiple Buffer Transfer Protocol (MBTP), a natural extension of the Simple Buffer Transfer Protocol (SBTP), is essentially the merging of multiple copies of SBTP to form a protocol that uses multiple channels for data transfer. However, instead of using both multiple senders and multiple receivers, MBTP uses a single sender that coordinates with a user-specified number of receivers. The protocol works as follows. The sender notifies each receiver that it is ready to send, and then waits until a receiver indicates that it is ready to receive. The sender selects one such receiver, and transmits a complete buffer to that receiver. While this buffer is being written to disk, the sender selects another receiver that has indicated it is ready to receive, and transmits a buffer to that receiver. This process continues until the complete file is sent.

This protocol maximizes the amount of time that the network is active. Ideally, while one receiver is writing to disk, another will be ready to receive more data from the sender.

We conducted an experiment to determine the effectiveness of the Multiple Buffer Transfer Protocol. We transferred a 25 Megabyte file between all possible workstation pairs, using both MBTP and ftp. For MBTP we specified 5 receivers, a buffer size of 1 Megabyte, and a packet size of 9000 bytes. All data transfers were conducted during a single working session, to make the comparisons as valid as possible. Results, measuring throughput in Mbps, are presented in Table 3. Note that it is impossible for us to control, or even to measure, background traffic on BAGNet, since other CalREN testbeds share the same Pacific Bell ATM switches and trunks. It should also be noted that we have made no attempt to determine an optimum number of receivers or an optimum buffer size for MBTP. Clearly, these are numbers that will be dependent on the particular source/destination pair and the state of the network during the data transfer. Because of the relatively low throughput values in Table 3, we suspect that we were experiencing contention for network resources.

During our experimentation with MBTP, throughput results have varied significantly. Some of our highest throughput values include 60 Mbps for MBTP versus 12.9 Mbps for ftp from *conan* to *chips* and 56.3 Mbps for MBTP versus 13 Mbps for ftp from *dufus* to *conan*.

5. Conclusions and Future Work

The way to maximize network throughput is to keep the network busy, and to keep it busy doing useful work. Two different approaches to achieving this goal are reported in the literature. The first approach is to keep

Table 3: Comparison of Throughput (Mbps) using FTP and MBTP

	chips	*conan*	*dufus*	*psyche*
chips	---	ftp: 15.4 MBTP: 17.5	ftp: 2.1 MBTP: 18.6	ftp: 2 MBTP: 20.5
conan	ftp: 20 MBTP: 32	---	ftp: 5.3 MBTP: 21.4	ftp: 6.9 MBTP: 15.4
dufus	ftp: 16.7 MBTP: 33.3	ftp: 15.4 MBTP: 18.4	---	ftp: 11.8 MBTP: 20
psyche	ftp: 11.8 MBTP: 32	ftp: 12.5 MBTP: 18.4	ftp: 10.5 MBTP: 20	---

the communications pipe full by transferring large blocks of data; the second is to avoid retransmission by using some type of forward error correction.

Data-transfer protocols that have been specifically designed to keep the communications pipe full by sending large bursts of data include NETBLT [4], BTOP (Bulk Transfer Oriented Protocol) [3], and mftp*[5]. NETBLT runs over IP; BTOP is intended to run within the AAL5 layer over ATM networks; and mftp runs over TCP/IP. All these protocols provide for retransmission of lost packets. Pursuing the second approach to achieving high network throughput, Turner and Peterson [13] and Shacham and McKenney [12] propose encoding techniques that enable forward error correction. Williamson [14] investigates optimizing the rate of file transfer by using feedback from a loss-load model that measures network congestion. This approach would also minimize retransmission.

In this paper we presented an unreliable protocol for data transfer. Our Multiple Buffer Transfer Protocol combines both of the above approaches to maximizing network throughput. The results in Table 3 establish the feasibility of our approach. In the future we plan to improve the efficiency of the algorithm; to conduct tests to determine how to optimize protocol parameters, such as the number of receivers and buffer size; and to determine the impact of background traffic on protocol performance. Then we plan to test our protocol in the context of an image-transfer application. We will characterize the pattern of errors that occurs when we use MBTP, and we will investigate how to correct these errors by using the information contained in adjacent pixels of the transferred image.

* mftp, which was designed for use with AERONET (the NASA proprietary network providing access to the Numerical Aerodynamic Simulation supercomputer facility at NASA Ames), uses multiple ftp streams to overcome the window-size limitation of TCP on networks with a high bandwidth-times-delay product.

Acknowledgments. We wish to thank Jerry Friesen of Sandia National Laboratories - Livermore and David Wiltzius of Lawrence Livermore National Laboratory for allowing us to use their workstations to conduct our experiments.

References

1. "BAGNet Home Page," World Wide Web document, http://george.lbl.gov/BAGNet.html.
2. L. Berc, "Debugging an ATM packet loss mystery," World Wide Web document, http://chocolate.pa.dec.com/mystery/mystery.html.
3. L. Casey, "The Design of BTOP - An ATM Bulk Transfer Protocol," Proceedings of 4th International IFIP Workshop on Protocols for High Speed Networks, 1994, pp. 149-166.
4. D. Clark, M. Lambert, and L. Zhang, "NETBLT: A Bulk Data Transfer Protocol," RFC 998, 1987.
5. J. Hahn, "MFTP: Recent Enhancements and Performance Measurements," NAS Technical Report RND-94, 1994.
6. J. Heinanen, "Multiprotocol Encapsulation over ATM Adaptation Layer 5," RFC 1483, 1993.
7. M. Johnson, "Experiences with the Bay Area Gigabit Network Testbed," Proceedings of the Fifth

IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems, 1995, pp. 26-32.

8. R. Jones, "Netperf Homepage," World Wide Web document, http: / / www.cup.hp.com / netperf / NetperfPage.html.

9. M. Laubach, "Classical IP and ARP over ATM," RFC 1577, 1994.

10. K. Moldeklev and P. Gunningberg, "Deadlock Situations in TCP over ATM," Proceedings of 4th International IFIP Workshop on Protocols for High Speed Networks, 1994, pp. 219 - 235.

11. A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks," Proceedings of ACM SIGCOMM '94, 1994, pp. 79-88.

12. N. Shacham and P. McKenney, "Packet Recovery in High-Speed Networks Using Coding and Buffer Management," Proceedings of IEEE INFOCOM '90, 1990, pp. 124-131.

13. C. Turner and L. Peterson, "Image Transfer: An End-to-End Design," Proceedings of ACM SIGCOMM '92, 1992, pp. 258-268.

14. C. Williamson, "Optimizing File Transfer Response Time Using the Loss-Load Curve Congestion Control Mechanism," Computer Communication Review, Vol. 23, No. 4, 1993, pp. 117-126.

15. D. Wiltzius, "LLNL ATM Performance Data," WorldWide Web document, http: / / www. llnl. gov/bagnet/perf-llnl.html.