Application-Level Congestion Control Mechanisms for Large Scale Data Transfers Across Computational Grids

Phillip M. Dickens and Vinod Kannan
Department of Computer Science
Illinois Institute of Technology

Abstract

In this paper, we discuss the development of a highly efficient, application-level data transfer system for computational grids. We describe and evaluate two application-level congestion control mechanisms, and compare their performance with respect to each other and with that obtained by TCP. We show that both application-level protocols are able to obtain performance that is very close to the maximum available bandwidth while keeping data loss rates low (from 0.06% to 1.6 %). Also, we show that both approaches obtain throughput that is close to an order of magnitude greater than that achieved by TCP. Finally, we begin to address the important issue of whether all data loss should be assumed to represent a network congestion event, with the goal of crafting the response to observed data loss as a function of the root causes of such loss.

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 [3, 8][1]. The reason for such poor performance is that the Transmission Control Protocol (TCP), 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 [2, 8]. This issue has led to research aimed at improving the performance of the TCP protocol itself in this network environment [2], as well as developing application-level techniques that can circumvent the performance problems inherent within the protocol [6, 9, 13].

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 [1], RUDP [9, 10], and previous versions of FOBS [5, 7] 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

[1] Please note that due to space limitations the reference list is representative rather than comprehensive.

(taken by SABUL [13]) 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).

Our research is focusing on the development of effective and efficient congestion control mechanisms that are appropriate for application-level data transfer mechanisms operating in a high-bandwidth, high-delay network environment. It is our belief that operating from an application perspective provides many opportunities for enhanced performance and enhanced resource utilization. For example, operating at the application level makes possible the viewing of the problem from a *system-level* perspective, where information related to the state of the end-to-end system can be collected and integrated into the congestion control mechanism. Similarly, the application can collect, analyze, and incorporate into the control mechanisms a wealth of historical information related to both intra- and inter-session data transfers. It is also possible to develop *families* of control mechanisms, where the choice of the controller is based on the characteristics of the end-to-end system. It would also be possible to dynamically switch between control mechanisms based on current system conditions.

In this paper, we present results of our initial exploration of the very rich solution space that becomes available when controlling data transfers from the application level. In particular, we implement and evaluate two different control mechanisms for large-scale data transfers across high-performance, wide area network connections. Both approaches attempt to minimize packet-loss but respond to such loss in very different ways. One mechanism uses what may be termed *state-based congestion control* where control decisions are a function solely of the current state and the current loss rate. This is an aggressive protocol that attempts to capture all otherwise unused bandwidth in a manner that is consistent with maintaining a low level of packet loss. The second approach is a non-aggressive protocol that attempts to find and remain within a sending range that allows it to maintain a data loss rate very close to zero.

There are three primary contributions of this paper. First, it clearly demonstrates that application level congestion control mechanisms can obtain a very high percentage of the available bandwidth while maintaining a very low data loss rate. Secondly, it shows that it can be quite useful to develop a family of congestion control mechanisms, each of which is targeted to a particular (expected) network environment. Finally, it begins to explore the important issue of the root causes of observed data loss, and raises the question of whether all data loss represents a debilitating network congestion event.

The rest of the paper is organized as follows. In Section 2, we provide an overview of the FOBS data transfer mechanism. In Section 3, we discuss the characteristics of the two protocols studied. In Section 3, we discuss the experimental design, and we provide the results of this experimentation in Section 4. In Section 5, we discuss related work, and we provide our conclusions and current research directions in Section 6.

2. The FOBS Data Transfer System

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 [5-7, 14]. 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 System-Aware FOBS

System-aware FOBS 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 instilling them with the ability to collect and share end-to-end system-state information is a focus of current research. The data transfer engine itself is (at least conceptually) reasonably simple, and details of its implementation are provided in [4-7]. For this discussion, all that is required is a basic understanding of the reliability mechanism since it is closely connected to the congestion control mechanisms.

FOBS employs a simple acknowledgment and retransmission mechanism where the file to be transferred is divided into data units we call *chunks*. Data is read from the disk, transferred to the receiver, and written to disk in units of chunks. 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, and this number was also chosen in consideration of the MTU size. These bitmaps are sent from the data receiver to the data sender at intervals dictated by the protocol, and triggers (at a time determined by the congestion/control flow algorithm) a retransmission of the lost packets. The bitmaps are sent over a TCP socket.

2.2 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 to provide effective congestion control and ensure fairness. One problem with this approach is that a small spike in packet loss may 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. Similarly, this approach can result in significant variance in offered load if the control mechanism iterates between increasing the sending rate due to low packet loss, observing a spike in packet-loss due to the increase in the sending rate, and then 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 are linear. The other approach is *state-based*, where the control interval is decreased from a chunk to a segment, and the increase/decrease parameters are functions of the current state and current feedback. We discuss each approach in the following sections.

It is important to note that historical knowledge is incorporated into the first protocol in three important ways. First, it was observed 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). Secondly, it was observed 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 safe sending rate was quite often close to S. This implies that the decrease parameter for the protocol does not need to be large.

2.2.1 Extending the Feedback Control Interval

Currently, the congestion control algorithm uses packet-loss rates to make decisions about modifying the behavior of the system. Thus, in its current form, FOBS uses the same feedback information as TCP and TCP-like algorithms. (We are however pursuing the collection and introduction of system-level information into the control mechanism.) One difference however is that decisions regarding changes in system behavior are made after one complete *chunk* has been successfully transferred. The other difference is that both the increase and decrease parameters are linear. The basic algorithm is as follows.

The mean loss rate for the entire chunk is calculated once the data has been successfully transferred, and the sending rate is reduced if this rate exceeds a given threshold (currently ½ of 1%). If this threshold is exceeded, the sending rate is reduced by a constant 5 Mbs (if the slowest link is 100 Mbs). This decrease rate was chosen based on extensive experimentation, and we are currently investigating techniques that allow the control agents to choose and (possibly modify) this value dynamically. Thus the decrease parameter may be modified, but the feedback-control interval remains constant.

The increase parameter is also linear, but the algorithm does *not necessarily increase* the sending rate each time the loss-rate falls (or remains) below the threshold value. This feature of the algorithm reflects the assumption that, at least for some network connections, there is a reasonably tight band in which the protocol can safely operate and increasing the sending rate past this upper bound will (or can) result in non-trivial packet loss. The bounds of these so-called safe operational bands may change during the execution of a large-scale data transfer, but it is assumed (and observed) that such changes are, in large part, neither frequent nor severe. That is, when such boundaries change, the new boundaries tend to remain fixed long enough for the algorithm to detect them. This aspect of the protocol is implemented by tracking the number of attempts made to move into a higher operational band, and the number of times such a move has resulted in a significant increase in packet loss. If, based on such historical information, it is deemed that an increase in sending rate will result in a similar increase in packet loss, the rate will remain unchanged.

2.2.2 State-Based Rate Changes

We are also investigating what may be termed a *state-based* approach where changes in the behavior of the system are based solely on the current state and current feedback information. We have implemented three states thus far: Green, Yellow, and Red. These states respectively represent excellent, moderate, and poor system conditions. Each state has its own minimum and maximum sending rates and amounts by which this rate can be increased or decreased, as well as a distinct set of conditions under which a state change will occur. To increase the responsiveness to system conditions the state is re-evaluated after a segment of data has been successfully transferred. The algorithm will significantly (and quickly) increase or decrease its sending rate if there is either negligible or massive packet loss respectively.

3. Experimental Design

We tested the performance of the application-level protocols and TCP between sites connected by the Abilene backbone network. One connection was between the Center for Advanced Computing Research (CACR, California Institute of Technology), and Argonne National Laboratory (ANL). The other connection was between CACR and National Center for Supercomputing Applications (NCSA, University of Illinois at Urbana-Champaign). The bottleneck on the link between CACR and ANL was a 100 Mbs interface between ANL[2] and Abilene. Similarly, the maximum rate at which we were able to place data

[2] The Gigabit Ethernet connection between the machine used in these experiments at ANL and the Abilene backbone network experienced severe hardware failures as we were performing our experiments. The Gigabit Ethernet card associated with the previous machine was replaced by a 100 Mbs interface card on the (temporary) replacement system.

onto the network from CACR to Abilene (without causing massive data losses) was also 100 Mbs[3]. The hardware and software configuration for these machines is provided in Table 1.

We transferred 50-Gigabytes of data between CACR and ANL, and between CACR and NCSA, for both application-level control mechanisms. We tested the performance of TCP across these same links, but did so for a much smaller data set (40 MB) since this was sufficient to establish the performance characteristics of the protocol. It is important to note that we were unable to increase the TCP buffer sizes since on these hosts kernel-level permission is required to do so. However, the comparison is still valid in the sense that one significant advantage that comes with operating at the application level is an essentially infinite transmission window.

4. Experimental Results

The results of these experiments are shown in Table 2. As can be seen, both application-level protocols performed quite well on the link between CACR and ANL where each obtained over 90% of the maximum available bandwidth. The primary difference in the performance of the two protocols is in the level of packet loss. The non-aggressive approach was able to maintain a loss rate of 0.06% while the more aggressive protocol had a loss rate of 1.6%. The reason for these results has to do with the characteristics of both the network connection and the communication endpoints. In particular, the receiving host (ANL) is generally lightly loaded, as is the network connection between ANL and CACR. Thus the aggressive protocol tends to reach and exceed the maximum sending rate that can be supported by this connection. The non-aggressive protocol on the other hand tends to find and stay at a sending rate just below this maximum threshold. TCP performed quite poorly on this connection, obtaining only 7% of the available bandwidth.

Both application level congestion control mechanisms also performed well on the connection between CACR and NCSA. The state-based control mechanism achieved a throughput rate of over 90% of the maximum available bandwidth with a loss rate of 1.15%. The non-aggressive protocol however was able to achieve only 82% of the maximum available bandwidth with a loss rate of 0.2%. TCP on the other hand was able to achieve a throughput rate equal to only 15% of the maximum available bandwidth.

The results obtained by the application-level protocols can again be explained by the characteristics of the end-to-end connection. In particular, the host at the receiving end (NCSA) is lightly loaded for reasonably long periods of time, but the system load tends to spike up on an infrequent and (generally) short-lived basis. As discussed below, we speculate that these spikes in system load result in similar spikes in packet loss. The state-based approach recovers quickly from spikes in the packet loss rate resulting in a higher overall throughput rate. The non-aggressive approach however tends to keep reducing its sending rate in search of a "safe" range within which it can transmit data (with a very low loss rate), and increases this sending rate very slowly.

It is important to note that the packet loss rates shown in Table 2 represent the mean values taken over the entire 50-Gigabyte data transfer. It is also important to consider the fluctuations in the loss rate over the life of the transfer, and to speculate on the impact of such fluctuations. We discuss these issues in the next section.

4.1 Spikes in Packet Loss

In both application-level approaches the packet loss rate remains very low for large segments of the data transfer. However, there are periods within which the loss rate spikes up (very briefly) to close to 50% for the state-based approach, and up to 20% in the non-aggressive approach. The question then is whether such spikes represent increasing contention within the network (thus causing other applications to similarly loose data), or whether it represents some other, perhaps less critical, event in the end-to-end system. We are very interested in this issue since it has a direct bearing on the question of whether large-scale data

[3] The reason for this poor performance is not yet understood as the documentation shows a Gigabit Ethernet connection between the front end and Abilene. We are attempting to resolve this issue with the system administrators at CACR.

transfers are more appropriately viewed from an end-to-end system- or network-level perspective. While it is difficult to answer such questions definitively, we can look to see if a change in the state of some other system component coincides with changes in the rate of observed packet loss.

To investigate this issue, we performed a 50-Gigabyte data transfer between CACR and NCSA with the sending rate set to a constant 95 Mbs and the congestion control mechanisms disabled. This allowed the packet loss rate to fluctuate solely in response to changing end-to-end system conditions. Concurrently, we used the System Activity Reported (SAR) to track changes in system load at the receiving host. The idea is that as the load on the system increases, the amount of CPU time available to the receiving process decreases. Thus the receiving process will more often be either swapped out or waiting in the run queue when the UDP data packets arrive at the network interface, and many such packets will be discarded due to the best-effort nature of connectionless protocols. The results of these experiments are shown in Figures 1 and 2.

As can be seen, the spikes in data loss appear to correspond quite closely with spikes in system load at the receiving host. While we have not yet performed a detailed statistical analysis of this apparent relationship (although we are currently conducting such a study), it is certainly reasonable to speculate that such a relationship exists. Assuming this is in fact the case, the next question is how to use this type of information in the congestion control mechanisms. Our initial response is to say that if the spikes in observed data loss *do not in fact* represent congestion developing within the network, then the existence of such spikes is perhaps not significantly problematic (at least with respect to the network). Thus it may be reasonable to tolerate short-lived increases in data loss if it can be determined that (at least with high probability) it is not going to result in congestion collapse in the network. However, the impact on the receiving host also has to be determined and considered in the congestion control mechanisms.

5. Related Work

There has been a significant amount of research related to the development of application level communication protocols including RUDP [9], SABUL [13], GridFTP [1], and PSockets [12]. Of these approaches, RUDP does not implement congestion control, and GridFTP and Psockets essentially circumvent the TCP congestion control mechanisms by allocating multiple TCP streams for a single data transfer. SABUL does implement congestion control that is closely aligned with that employed by TCP. In particular, it employs a feedback control interval on the order of a single round trip time (similar to TCP), and reduces its sending rate by a constant amount when the loss rate exceeds a given threshold. Also, there has been research related to modifying the congestion control parameters within TCP itself to provide better performance in a high-bandwidth, high-delay network environment [11].

FOBS is unique in that it is developing multiple congestion control mechanisms with the ability to dynamically switch between mechanisms in response to changes in the state of the end-to-end system. Also, we are investigating approaches that allow for a much longer feedback control interval (i.e. significantly greater than a single round trip time) while maintaining very low rates of data loss. Finally, we are investigating mechanisms to determine the root causes of observed packet loss, with the goal of providing responses that are tailored to the particular problem being encountered.

6. Conclusions and Current Research

In this paper, we have discussed the initial phases of our research into application-level congestion control mechanisms for a high-performance, application-level data transfer system. We have shown that both approaches are able to achieve between 80% and 90% of the maximum available bandwidth, while maintaining an overall loss rate between 0.06% and 1.15%. This compares quite well with TCP that was able to achieve only 15% of this maximum rate. Also, we have presented experimental results in combination with system monitoring to push forward the notion that large-scale data transfers are more appropriately viewed from a system, rather than a network, perspective.

Our current research is aimed at further developing the concept of system-aware data transfers, where information related to the state of the end-to-end system is collected and leveraged in the congestion

control mechanisms. Towards this end, we are developing sensors to monitor the state of various system-level components, and performing statistical analyses to explore the relationships between changes in the end-to-end system and the behavior and performance of the data transfer protocol.

References:

[1] 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.

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

[3] 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

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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).*

[13] 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*.

[14] 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.

TABLES:

Host	Architecture	NIC	Operating System
ANL	2* 500 MHz Pentium 3.	100 Mbs	Linux Red Hat 7.3
NCSA	IBM IntelliStation Z Pro 6894 (dual processor)	Gig-E	Linux Red Hat 7.1
CACR	HP N4000 4 processor	100 Mbs	HP-UX 11.10

Table 1: This table depicts the hardware and software configurations of the hosts used in this study.

Network Connection	Protocol	Percentage of Maximum Throughput Achieved	Overall Packet Loss Percentage
CACR to ANL	Non-Aggressive	92%	0.06%
CACR to ANL	State-Based	93%	1.6%
CACR to ANL	TCP	7%	———
CACR to NCSA	Non-Aggressive	82%	0.2%
CACR to NCSA	State-Based	92.6%	1.15%
CACR to NCSA	TCP	15%	———

Table 2: This table depicts the percentage of the maximum possible bandwidth obtained by the two application-level protocols as well as the loss rates for the various network connections. Also, it shows the percentage of the maximum bandwidth obtained by TCP over these same connections.

FIGURES:

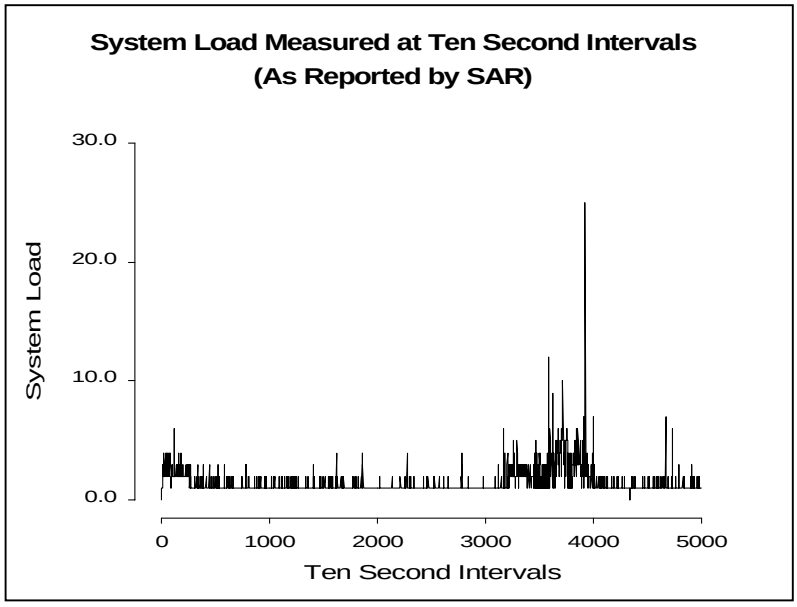

Figure 1: This figure depicts the system load at ten second intervals over a 13-hour period as reported by SAR. The host was User01 at NCSA.

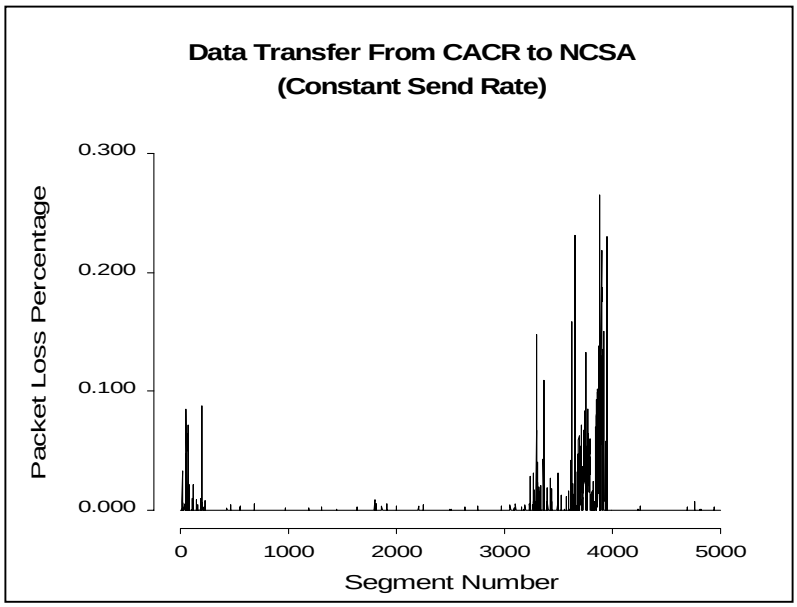

Figure 2: This figure depicts the packet loss rate over the same 13-hour period.