TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTE

The work presented in this thesis has been submitted as two papers to 16th International Conference on Parallel and Distributed Computing Systems.

ABSTRACT

Advances in networking technologies have led to the deployment of networks capable of sending at up to 40 gigabits per second. Concurrently, research on Grid computing, has facilitated collaboration among geographically distributed researchers. This confluence of Grid-based computing and networking technologies is making possible high-performance applications to solve massive problems in a variety of fields, including genome research, geophysics, astronomy and astrophysics, tele-immersion, and medicine. Experience with existing computational Grids shows, that applications are able to only utilize 10 percent or less of the available bandwidth. This is because TCP (Transmission Control Protocol), the default data transfer protocol, is not designed for high-bandwidth, high-delay network environments.

Our alternative, called FOBS, (Fast Object Based System), is a highly efficient, lightweight application level data transfer mechanism developed for high-bandwidth, high-latency networks. FOBS is an application level mechanism that uses User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). Because it is UDP based, reliability has to be built in at the application-level, including an acknowledgment and re-transmission mechanism. FOBS has been shown to achieve up to 90% of the available bandwidth across both long and short haul networks. However, early versions of FOBS were not able to scale to very large data sets and higher bandwidth networks, and bandwidth utilization was not the best possible. Two factors were responsible. First, the acknowledgment mechanism did not perform well as the size of the data sets increased. Second, the early versions did not implement any form of rate or congestion control.

These two factors made it impossible for FOBS to be deployed for transferring large data sets in a non-dedicated environment, as the packet loss was unacceptably high.

This thesis describes the design and technical issues were addressed to overcome the critical deficiencies in the initial FOBS system. In particular, the thesis describes various techniques for acknowledgment and retransmission. The rationale for the changes is also explained, and it is shown that the best technique performs well across all the networks tested. The result of this aspect of research is a user-level acknowledgment and retransmission mechanism that provides excellent performance over any data set size and any bandwidth. The adopted design is also shown to scale to higher bandwidth high performance networks. The size of the data set and the cost of acknowledgment and retransmission have been decoupled. FOBS now utilizes up to 95 percent of the maximum.

This thesis also describes a new rate and congestion control mechanism, which is shown to keep the data loss between 0 and 2 percent, well within acceptable boundaries. As a result of this research, FOBS is now a highly efficient, highly scalable application-level mechanism usable in generic high-performance networks.

CHAPTER I

INTRODUCTION

The national computational community is moving toward a Grid-based model wherein high-speed, high-bandwidth wide-area networks (WANs) link geographically distributed computational resources [28, 13]. Efforts have been made, both commercially and by the scientific research community, to lay the backbone for the next generation of the Internet [Internet2] and for scientific research networks [28, 1, 26]. Already, commercial and research networks provide up to 10 gigabits per second of connectivity [1]. Backbone networks capable of delivering data at 40 gigabits per second [28] are being laid to connect the distributed computational resources. The goal is to provide computational resources on demand and to “commoditize” computation. This so-called computational Grid has made possible large-scale, data-intensive distributed applications that were previously infeasible.

At the heart of any such Grid environment is the ability to transfer very large amounts of data in a highly efficient manner. All of the developing and envisioned advanced distributed applications are predicated on this fundamental ability. It has been well established that in practice, the actual bandwidth achieved by existing distributed applications executing in a Grid environment, such as the Internet2 infrastructure, represents only a very small fraction of the available bandwidth [1, 9].

1.1 Motivation for the Research

Existing protocols used for data transfer, do not perform well in a high-performance environment. The reason is that TCP (Transmission Control Protocol), the data transfer mechanism of choice for wide-area data transfers, performs poorly in a high-bandwidth, high-delay network environment [11, 21, 15, 16]. TCP's poor performance can be attributed to two primary reasons:

1. The size of the TCP congestion control windows.
2. The inordinately aggressive congestion control mechanism, which was designed primarily for the highly congested, unreliable low-performance Internet1 environment.

Given the performance problems inherent in TCP, a significant amount of research is aimed at developing more effective techniques for delivering data across high-performance computational Grids.

Solution to the problem involves two aspects:

- Increasing the bandwidth utilization or throughput achieved.
- Developing a congestion-control model more suited to high-bandwidth WANs.

Current research has taken two approaches to solving the problem of low utilization of installed bandwidth. In one, research has focused on improving the performance of TCP, making it more suitable for high-bandwidth, high-latency networks [10, 11, 12]. In the other, researchers are developing application-level techniques to circumvent the performance problems associated with TCP [2, 5, 6, 25, 17, 24].

In the former approach, the size of the TCP window is the single most important factor in achieving good performance on high-latency, high-bandwidth networks. Such “fat” pipes are kept full by increasing the TCP congestion window size to at least the product of bandwidth and the round-trip delay. Thus, research has focused on automatically tuning the size of the TCP socket buffers at run time. Commercial TCP implementations have been developed that allow the system administrator to significantly increase the size of the TCP window to achieve better performance. Another area of active research is the use of selective acknowledgment mechanism. Here, a selective acknowledgment (SACK) packet that specifies exactly those packets that have been received is sent from the receiver to the sender, allowing the latter to retransmit only those packets that are missing.

In the latter approach, early solutions at the application level have focused achieving significantly better bandwidth utilization. One way to achieve better utilization has been to circumventing the performance flaws in TCP by allocating multiple TCP streams for a given data flow [6, 2]. This approach can provide significant performance enhancement because the limitations of the TCP window sizes are on a per socket basis and, thus, the striping of the data across multiple sockets provides an aggregate TCP buffer-size that is closer to the ideal size. Moreover, this approach essentially circumvents the congestion-control mechanisms of TCP. That is, while some streams may be blocked as a result of the congestion control mechanism, it is likely that others are ready to fire. The larger the number of TCP streams, the lower the probability that all such streams will be blocked.

Two other approaches at the user level, closely related to FOBS are Reliable Blast User Datagram Protocol (RUDP) [17] and Simple Available Bandwidth Utilization Library (SABUL) [25]. RUDP works on the entire data set at one time and is designed for high-performance quality-of-service (QoS)-enabled networks with a very low probability of packet loss. SABUL [25] employs a single UDP stream for data transmission and a (single) TCP stream for control information related to the state of the data transfer. But their congestion control has been closely tied to that of TCP.

The need remains for an acknowledgment and retransmission mechanism that is scalable over large data sets and "fat", high-latency pipes. Also needed is a congestion control model for these high-performance networks for transferring large data sets. The initial approaches such as [22, 25] and [17] did not provide a congestion control mechanism or provided one that was too aggressive and closely aligned with TCP. At the same time, another important aspect of the Grid is that hosts of various architectures and operating systems come together to form coherent virtual organizations. Hence, performance should be decoupled from system architectures or operating system characteristics. The SABUL approach has been to utilize operating system and architecture specific functionalities to get the peak performance. This is not a practical solution given the heterogeneous nature of the hosts on the Grid.

Security is also a key consideration while creating and deploying high-performance applications on the Grid. Traditional approaches either have provided little or no security. FOBS approaches security at two levels: the system startup and the system access levels.

This thesis focuses on overcoming the shortfalls of the previous approaches. The goal is to design and implement a user-level acknowledgment and retransmission mechanism that scales up in terms of the bandwidth, latency, and capabilities of the end hosts. This thesis describes the development of a technique that increases bandwidth utilization. The thesis presents a congestion model that shares high-delay WANs fairly with existing protocols. This thesis also explains the FOBS security aspects and explains the mechanisms used to ensure secure access and use of FOBS. The work presented in this thesis has led to a high-performance, scalable, portable, and secure data transfer mechanism that is easy to deploy and use.

1.2 Organization of the Thesis

The remainder of this thesis is structured as follows. Chapter 2 presents related work and previous solutions to the transfer protocol problem. Traditional approaches are analyzed and explained. In Chapter 3, an overview of the FOBS approach is presented. The initial approach and the drawbacks faced are explained. The redesigned FOBS and the rationale for the design changes are described. The components of FOBS, including the acknowledgment and retransmission mechanism, the congestion-control mechanism and portability issues are explained. The rationale behind the design of each component and its impact on performance are explained. Scalability of the FOBS acknowledgment and retransmission approach is studied both in terms of the bandwidth of the network links on which FOBS is deployed and in terms of the processing capabilities of the end hosts. In Chapter 4, the experimental setup and the test results quantifying the impact of the design are presented. The FOBS protocol is shown to be scalable to high-bandwidth networks. Chapter 5 summarizes the research and recommendations for future work are made.

CHAPTER II

RELATED WORK

This chapter reviews research work relevant to the scope of this thesis. We identify the key aspects that differentiate our research from the related work reviewed in this chapter.

Current research aimed at solving the poor bandwidth utilization on high-performance networks can be classified under two broad categories based on the approach taken by them to address the problem. They are improvements to the TCP and user-level approaches.

2.1 Improvements to TCP

TCP is the default protocol for data transfer on both Internet1 and high-performance networks. Various data transfer applications, such as file transfer protocol (FTP) and secure copy protocol (SCP), use TCP as the underlying mechanism to ensure the reliable delivery of data sent across the network.

TCP relies on two mechanisms, flow and congestion control, to set its transmission rate. Flow control endures that the sender does not overrun the receivers available buffer space; congestion control ensures that the sender not unfairly overrun the network's available bandwidth [16, 11]. The implementation of these mechanisms is through a flow-control window (fwnd) and a congestion control window (cwnd). The effective window (ewnd) is calculated as ewnd = min (fwnd, cwnd), and data is then sent at a rate of ewnd/ Round trip time (RTT), where RTT is the round-trip time of the connection. Cwnd varies dynamically as the network state changes, while fwnd is always static. Ideally, fwnd should vary with the bandwidth delay product (BDP), a measure of the amount of data in transit [11].

A static fwnd has been sufficient on networks with a low BDP. Fwnd was set to obtain acceptable performance while wasting little memory. On most operating systems, fwnd is set to 64 KB without scaling [16]. But the BDP can vary from a few bytes (56 Kbps $*$ 5 ms = 36B) to a few megabytes (1000 Mbps $*$ 100ms = 7.8 MB). Thus in the first case, over 99% of the memory allocated is wasted, while in the latter, 99% of the network bandwidth is wasted. Over the span of the network transfer of a massive data set (in order of gigabytes and terabytes), the network delay and thus the BDP change, because of transitory queuing and congestion. Thus, a fixed fwnd is not ideal. Selecting a fixed window size will underallocate memory and underutilize the network, when the BDP is as large as on the high-performance networks.

TCP also suffers from problems other than the window size, including self-similar (chaotic) behavior [3, 29]. In TCP, the acknowledgment is for each segment transmitted, on a cumulative basis [19]. This leads to considerable overhead and degradation of the performance of applications using TCP for data transfer.

Thus, in summary, TCP has been proven to perform poorly in a high-bandwidth, high-delay network environment [11, 21, 15, 16]. This is because of the inherent characteristics of the TCP, which include

1. The window-based congestion control mechanism and the size of the window.
2. The aggressive congestion control mechanism, which is designed for the Internet1.
3. The slow start mechanism, which is necessary for low-performance Internet1 environment.
4. The acknowledgment scheme in basic TCP.

A considerable amount of research has focused on overcoming the above drawbacks. Active research has focused on addressing the congestion control problems in TCP [23, 7]. Another area of research has been the use of a selective acknowledgment mechanism [19, 10, 18], rather than the standard cumulative acknowledgment scheme. Here, the receiver specifies exactly which packets have been received, allowing the sender to send only those segments that have not been acknowledged. Additionally, “fast transmit” and “fast recovery” mechanisms have been developed that allow the TCP sender to retransmit a segment before the retransmission timer expires. These mechanisms also allow the TCP sender to increase the size of its transmission window when duplicate packets are received [19, 18]. Dynamic right sizing is also an approach to over come the problems of a static transmission window [11].

In spite of all the above efforts to overcome the drawbacks of TCP, significant problems persist. Research is thus ongoing to find an ideal solution. But, changes to TCP mechanism, usually involves root permissions to adjust the transmission window size. This in itself creates a host of problems. Thus, another category of research is focused on overcoming TCP's inherent problems on high BDP networks at the user level.

2.2 User Level Approaches

Various methods have been studied to develop application-level solutions to the problems discussed in the preceding sections. Research in this category has aimed either to circumvent the TCP's congestion control mechanism by striping the data across multiple TCP streams [24, 2] or to develop a completely fresh user level acknowledgment and retransmission mechanism [8, 9, 25, 27]. These approaches are discussed below.

2.2.1 PSOCKETS, BBCP, BBFTP and GridFTP

PSOCKETS [24], BBCP [5], BBFTP [6] and GridFTP [2] are solutions based on utilizing multiple TCP streams that in effect increases the size of the TCP window. The data to be transmitted is "striped" across multiple sockets that provide an aggregate TCP buffer size that is closer to the BDP. Using multiple streams also increases the probability that, at any given time, at least one TCP stream will be ready to fire. A similar approach has been investigated within the domain of satellite based information systems [20]. BBCP, BBFTP and GridFTP also take advantage of the research done to increase the performance of individual TCP streams as discussed in Section 2.1.

2.2.2 RUDP, TSUNAMI, and SABUL

RUDP, Tsunami and Sabul are closely related to FOBS. These approaches have aimed to develop a completely fresh user-level acknowledgment and retransmission scheme. Data is sent on UDP; and the control information, including acknowledgment, is sent across on TCP.

In RUDP, all the data is sent across the network on UDP. A message is sent to the receiver at the end of transmission, and after some timeout period the receiver sends a list of missing packets. The data sender then retransmits all of the lost packets. This process is iterated until all of the data has been successfully transferred. RUDP is designed for high-performance quality-of-service enabled networks with a very low probability of packet loss. RUDP does not provide a congestion control mechanism however and thus is unsuitable for shared networks.

SABUL employs a single UDP stream for data transmission and a (single) TCP stream for control information related to the state of the data transfer. SABUL uses system architecture and operating system specific function calls to deliver impressive performance. It uses techniques such as “User-Buffer Registration” supported only by a limited set of Linux kernels, to significantly decrease the processing involved in the data transfer. This is also linked to the Pentium Pro architecture and uses a high-resolution timer that is supported only on a very limited number of compilers. [25] is not portable across most of the systems. Considering the nature of the Grid, this is a critical drawback, which limits the use of this protocol to a very small set of hosts. Also, their congestion control has been closely tied to that of TCP. The sampling interval for congestion control is, inordinately small, especially when the network is essentially clear of any other traffic.

[27] is also similar to the above approaches. It is extremely aggressive, and experimental. It does not provide congestion control as needed for shared networks.

CHAPTER III

FOBS

In this chapter, we shall explain the initial design of FOBS as described in [8], and explain the shortcomings of this design. We shall also explain the redesigned FOBS protocol, to overcome the drawbacks in the initial design. The key components of FOBS, and the issues related to them will be explained. We shall also explain the issues related to the portability and usability of FOBS.

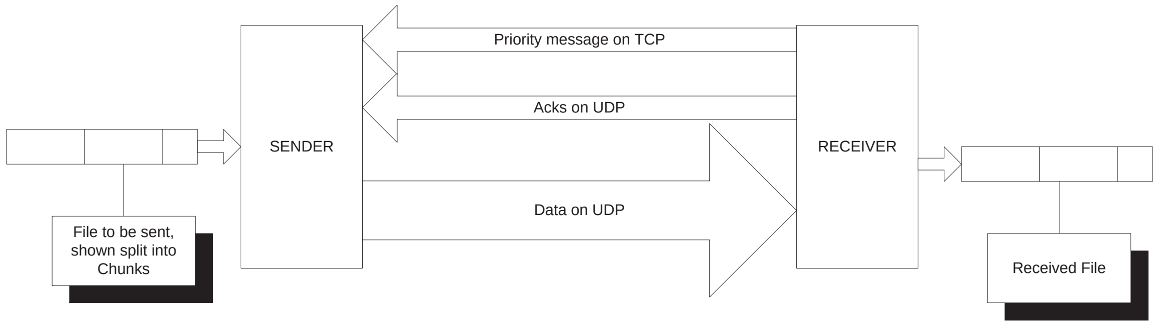
3.1 Overview of the FOBS data transfer mechanism

FOBS is a simple user level data transfer mechanism to be used for transfers of large data sets over high-bandwidth, high-latency network environments, like computational Grids. The design of the FOBS data transfer protocol, or for that matter any other application-level file transfer mechanism, has three main components: the sending algorithm, the receiving algorithm and the acknowledgment scheme. The choice of these components depends on factors such as the type of network, latency and CPU contention among others.

FOBS uses the User Datagram Protocol (UDP) as the data transport protocol. The performance implications of sending data using UDP are explained in Appendix A.1. Reliability is provided through an application-level acknowledgment and retransmission mechanism.

3.2 Initial FOBS design with Byte or Bit Acknowledgment

In the initial FOBS design, as explained in [8], files were transferred, by breaking them into smaller sized “chunks”. Data from the chunk was transmitted using UDP packets to the receiver. The feedback messages from the receiver were transmitted on UDP from the receiving side to the sending side and a TCP stream was used to pass the highest priority control messages signifying the complete receiving of a chunk.



```
ERROR: invalidrestore
OFFENDING COMMAND: restore

STACK:

-savelevel-
-savelevel-
```