THE UNIVERSITY OF CALGARY

A GridFTP Overlay Network Service

by

Philip Rizk

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

October, 2006

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "A GridFTP Overlay Network Service" submitted by Philip Rizk in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Dr. Brian Unger
Supervisor, Department of Computer Science

Dr. Rob Simmonds
Co-Supervisor, Department of Computer Science

Dr. Zongpeng Li
Department of Computer Science

Dr. Edward J. Vigmond
Department of Electrical and Computer Engineering

October 19, 2006

Date

Abstract

The Transport Control Protocol (TCP) is widely deployed and is used by the current implementation of Grid File Transfer Protocol (GridFTP) for data transfers within grid environments. Due to TCP dynamics, throughput of data transfers can be much lower than the available bandwidth in heterogeneous, high bandwidth networks. Splitting a TCP connection into two or more sequential segments has been shown to improve throughput on these types of networks.

This thesis presents a set of components that enable the deployment of overlay networks that make use of split-TCP connections to improve GridFTP transfer performance. The components include an extension to the Globus Toolkit version 4 GridFTP server that supports split TCP connections, a service to estimate bulk transfer capacity and a service to determine if and where to split a connection. Results of experiments presented demonstrate average throughput improvements in excess of 100% for connections using proxies are obtainable despite using very simple observations to determine the routing of the split connections.

Acknowledgments

I would like to offer my profound thanks to my supervisors, Dr. Rob Simmonds and Dr. Brian Unger for their support. Without the continued encouragement, financial support, and especially the guidance and mentoring of my supervisors it would not have been possible to complete this work.

I would also like to thank all the other members of the Grid Research Center. Their input always helped clarify my ideas and provided a stimulating intellectual environment. I am also very grateful to Dr. Cameron Kiddle who provided invaluable help in developing the ideas, and methods in this thesis and the related publications. Thank you to my committee members, Dr. Zongpeng Li and Dr. Edward J. Vigmond, for their excellent suggestions in improving this thesis. Finally, thanks to my friends and family for their support.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACK : Acknowledgment or Acknowledgment packet
ADSL : Asymmetric Digital Subscriber Line
AIMD : Additive Increase, Multiplicative Decrease
BDP : Bandwidth Delay Product
BTC : Bulk Transport Capacity
CDF : Cumulative Density Function
CFS : Cluster File System
DSI : Data Storage Interface
ECN : Explicit Congestion Notification
EWMA : Exponential Weighted Moving Average
FTP : File Transfer Protocol
GGF : Globus Grid Forum
GridFTP : Grid File Transfer Protocol GSI : Grid Security Infrastructure
HBDP : High Bandwidth Delay Product
HPC : High-Performance Computing
IP : Internet Protocol
IP-TNE : Internet Protocol Traffic and Network Emulator
ISO : International Standards Organization
LFN : Long Fat Networks
LSL : Logistical Session Layer
MDS4 : Monitoring and Discovery Service version 4
MSS : Maximum Segment Size
MTU : Maximum Transmission Unit
NFS : Network File System
NWS : Network Weather Service
OGSI Open Grid Services Infrastructure
OSI : Open System Interconnect PDF : Probability Density Function
PEP : Performance Enhancing Proxy
QOS : Quality of Service
RAM : Random Access Memory RED : Random Early Detection
RFC : Request For Comment
RFT : Reliable File Transfer
RON : Resilient Overlay Network
RTT : Round Trip Time
SABUL : Simple Available Bandwidth Utilization Library
SACK : Selective Acknowledgments
SCS : Split Choice Service
SSLv3 : Secure Socket Layer Version 3 protocol

TCP : Transport Control Protocol
UDP : User Datagram Protocol
URL : Uniform Resource Locator
WSRF : Web Services Resource Framework
XFTP : Extended File Transfer Protocol
XIO : eXtensible Input/Output

Chapter 1

Introduction

A clear trend in high performance computing (HPC) is the movement towards the Grid Computing paradigm; the presentation of computation, storage, networking and instrumentation resources as a set of services. Such a set of Grid services represents an HPC research infrastructure, or grid. The term "grid" is a result of a metaphor comparing the provision of resources to an electrical grid. Diverse sets of these resources from more than one administrative domain can be combined to quickly develop and deploy large distributed applications to address problems that were previously intractable. In order for this to occur, these services would ideally be as readily available as electricity is from an outlet.

The realization of a grid computing infrastructure requires the development of middleware that makes resources available and provides support services. Providing resources such as computation facilities and scientific instrumentation as services requires the development of well defined interfaces. Support services provide functionality that is required by most distributed scientific applications such as resource discovery, security, meta-scheduling, workflow, and data management.

This grid infrastructure is also placing increasing demands on the networking infrastructure. The increasing amount of instrumentation, from medical devices that allow doctors to observe patients remotely, to large sensor nets in the ocean, are often connected to the Internet via non traditional network technologies such as radio and ad-hoc wireless networking. These media have higher error rates. This results

in a increasingly complicated network topology that includes wireless and optical networks. The growing interest in on-demand networking with Quality of Service (QOS) guarantees will likely further complicate network technologies. As the number of instruments grow, the amount of data available to researchers also expands. Global consortia which pool databases, storage and computational resources mean the data and location of computation are farther apart. Meanwhile, the networking technologies being used to transport the data were not developed for these environments and can not meet data transfer speed requirements despite the physical infrastructure being theoretically capable of transmitting the data fast enough. For example the Transmission Control Protocol (TCP), the most commonly used protocol for file transfer assumes all loss is due to congestion, whereas many links have a high level of loss due to uncorrectable bit errors. TCP also has no mechanism to effectively take advantage of QOS guarantees. The gap between the data transport needs of the rapidly evolving grid infrastructure and current networking standards is widening.

An example of the current networking standards not meeting the needs of new grid applications is the ICE project. Data from sensors and cameras on the Confederation bridge is collected by a computer on the bridge and stored on magnetic tape drives. The tapes are then physically transported off the bridge to other locations for analysis. Ideally, the data would be transferred off the bridge automatically via the wireless link that connects the computer on the bridge to the University of Prince Edward Island. The software being used to gather the data on the bridge is a legacy application using an operating system that has limited choices for file transfer clients. This makes transporting the data from the site via the network difficult

despite the ability of the network to handle the data flow. The flow experiences a high bit error rate over the wireless link which is erroneously interpreted by the underlying transport protocol (TCP) as congestion and the send rate is needlessly throttled back. This results in transfer speeds that may be too low to keep up with the amount of data being generated despite the fact that each individual link is capable of transporting the data over noisy links at the necessary speeds.

The utilization issues TCP experiences in heterogeneous environments such as the ICE project occurs because network engineers generally adhere to to the principles espoused in the "end-to-end" argument [77]; the network should simply move data while any intelligence remains exclusively on the edge of the network. The Internet is based on best effort packet delivery and includes no mechanism for the network to communicate its state of congestion to the edges except by dropping packets. This means that the longer and more complicated the network becomes, the less capable the edges are of efficiently using the network.

It has been recognized for some time, that as networks become faster and more heterogeneous, TCP's ability to utilize the network will become worse. One way to to overcome TCP's limitations on these networks is to split a TCP connection into multiple shorter segments connected in series in order to improve performance. These segments are connected by proxies that provide additional buffer space and negotiate the TCP connections. This has generally been done for specific network topologies such as wireless [8, 10, 17], satellite [12], and hybrid Coaxial cable networks [20] because the high bit error rates, combined with high latency, make TCP very inefficient on these networks. The use of proxies to split a TCP connection in this manner was likely not considered in the general case because the routers would

require too much memory to carry out this duty for all traffic. It is also not appropriate for the network to make the decision of whether or not to use proxies because applications have different needs. Proxies are useful for improving throughput but can increase latency for example.

The development of grid services now make it viable in certain circumstances to make use of splitting a connection via a proxy at the application layer. Resource discovery services provide the necessary information to decide when and where to split a connection. New security protocols make it easier to manage access to proxies and new data transfer services can provide an efficient mechanism for using proxies. Splitting connections in this manner can be seen as making a proxy's network access available to other hosts. The work presented in this thesis seeks to provide the services required to split connections in order to improve file transfer performance. It provides the discovery, authentication, and resource management components required to use proxies to overcome network inefficiency that results from the latency and diversity in network resources. The components of the system form a Grid File Transfer Protocol (GridFTP) overlay network that enables GridFTP clients and servers to utilize split-TCP connection in GridFTP transfers.

It is possible that at some point in the future, other protocols that are more effective in transferring large amounts of data in wide area networks could be used in place of TCP. The Globus Toolkit v4 GridFTP server was developed with an eXtensible Input/Output (XIO) [3] library that makes it possible to replace underlying protocol modules for this very reason. However, the replacement of TCP with protocols that more effectively utilize bandwidth will take a considerable amount of time. A version of TCP's Selective Acknowledgements was first suggested in 1988

and the final RFC on which implementations are based is dated 1996 [59]. This eight year lag represents only one measure of the time from suggestion to an adopted RFC and does not consider the time it took for vendors to actually adopt and implement the option. This change represents an incremental change to an already existing protocol in which only the endpoints need to be modified. A protocol that replaces TCP would represent a much more significant change that likely includes more than just the edges of the network and therefore might take considerably longer. The proxy mechanisms developed in this thesis are an effective way to overcome TCP's inefficiencies while other protocols are being developed, tested and deployed. When new protocols are developed the proxy mechanisms developed here can still be used to make efficient use of on-demand bandwidth technologies. The mechanisms here could also be used to link legacy applications to more effective transport mechanisms.

The rest of this thesis is organized as follows. Chapter 2 provides background of various grid and networking technologies. TCP and its failure to effectively utilize bandwidth in certain networking environments is discussed. Chapter 3 provides a survey of previous work in improving TCP's performance and high performance data transfer. Chapter 4 presents the design and rational of the GridFTP overlay network. Chapter 5 presents both emulation and empirical performance results of the GridFTP overlay network. Chapter 6 provides a summary of the thesis and suggests future work.

Chapter 2

Background

This chapter presents background on the technologies relevant to the GridFTP Overlay Network Service. Grid technologies that provide security, resource discovery, data management, and data transfer are discussed. An overview of network protocols is given, with TCP more closely examined as it is the most widely used protocol for file transfer and the underlying protocol used by GridFTP. The GridFTP overlay network utilizes GridFTP protocol. The chapter concludes with an overview of overlay networks and other peer to peer (P2P) technologies.

2.1 Grid Computing

This section discusses some of the current grid computing technologies. These technologies are used to develop higher level services such as the GridFTP overlay network.

There have been efforts to arrive at common standards within the grid community since 1998 [37]. The Global Grid Forum (GGF) is a community of users, developers and vendors that worked towards this standardization. This eventually led to the Open Grid Services Infrastructure (OGSI) which defined a Grid service as a Web service that conforms to a set of conventions that define the interaction between a client and a Grid service [26]. In March of 2006, Hewlett Packard, IBM and Microsoft announced a commitment to develop a common set of specifications for stateful Web

service access, management, event handling and notification [34]. The combined clout of these vendors helped overcome obstacles towards the adoption of a common standard by various stakeholders. The Web Services Resource Framework (WSRF) that was developed essentially replaced OGSI and was adopted by most stakeholders as a standard on how to present Grid services as stateful Web services.

The majority of grid computing technologies used by the GridFTP overlay network are pieces of the Globus Toolkit version 4 (GT4) [33]. The Globus Toolkit has become the de facto standard for grid computing interfaces and the globus developers are committed to conforming to the WSRF standard [34]. The toolkit consists of a set of components providing low level functionality required by many grid services or distributed applications that make use of grid technologies. The functionality included in the Globus Toolkit includes security, resource discovery, data management and execution management. The types of services being developed are numerous and only the security, resource discovery and data management tools included in GT4 that are relevant to the GridFTP overlay network are discussed here.

2.1.1 Security

Grid Security Infrastructure (GSI) [35] is an X.509 certificate based credentialing scheme that provides authentication and authorization functionality. GSI uses a globally unique Distinguished Name (DN) to uniquely identify any subject. A subject may be a user, resource or program. A credential is any piece of information used to identify a subject [35]. GSI uses X.509 certificates as its credentials. The authentication algorithm is defined by the Secure Socket Layer Version 3 protocol (SSLv3). This scheme includes a signature algorithm that verifies a subject holds

a private key assumed to be accessible by only the subject. The most important feature of GSI is that any subject can create and sign a set of certificates referred to as proxies and delegate them to a process. This proxy is then used by the process to prove that they represent the subject in question. A process that holds a proxy can then use it to sign and delegate another proxy to another sub process. These proxy certificates form a chain of signatures up to the original signing certificate authority. The certificate authority is used as a trusted third party between any two subjects. So long as both subjects of an authentication exchange trust the same certificate authority, they can trust the validity of the credential certificate they are being presented with. The use of delegated proxies provides single sign on functionality allowing applications to integrate resources from multiple administrative domains.

2.1.2 Monitoring and Resource Discovery

GT4 includes a Monitoring and Discovery Service (MDS4) [79]. This suite of web services can be used to publish any information about a resource that may be meaningful. For example, an organization may want to advertise the capabilities of a cluster such as how many nodes are in the cluster, the memory available on each node, and how to submit jobs to the cluster. A job meta-scheduler can then use this information to discover resources that match a particular job's requirements. Libraries and client tools allow applications to easily publish and retrieve this information.

In addition to information for resource discovery, monitoring data is necessary for performance analysis, tuning and resource selection [43]. For example, an application choosing from several data sources to retrieve files from may wish to use information

about network load or available bandwidth in order select the source that will finish the transfer first.

MDS4 was designed to deal with the scalability issues inherent in grid environments. The system has a hierarchical structure allowing locally managed information sources to be easily aggregated. MDS4 has a soft consistency model; the data is not guaranteed to be the most recent. Any information registered with MDS4 also has a limited lifetime, ensuring that outdated information is automatically removed.

2.1.3 GridFTP

The GridFTP protocol is defined by the Global Grid Forum Recommendation GDF.020, RFC959, RFC2228 and RFC2398. It is an extension of the FTP protocol, providing features useful for performing high performance data transfers within a grid environment. These features include (but are not limited to) [2]:

- Grid Security Infrastructure (GSI).
- negotiated TCP buffer sizes.
- support for parallel transfers.
- support for partial and reliable file transfer through the use of restart markers that indicate what pieces of a file have been moved.

GridFTP is used on a large number of high performance computing sites because it is easier to attain high throughput transfers with it. The most important features contributing to this are the negotiated TCP buffer sizes and, in some environments parallel streams.

2.1.4 Data Management

Reliable File Transfer (RFT) [56] is a service that provides reliable transfer of files resilient to network and host computer failures. Without RFT, if a client fails or the host the client is on shuts down, the transfer must be restarted from the beginning unless the client saves the state of transfers itself. RFT accepts a list of file transfers to perform, along with a proxy certificate to use to authenticate, and then attempts to transfer the files. RFT makes use of GridFTP's restart markers and partial file transfer options to efficiently restart transfers that failed partway through. This state information is stored in a database so that the transfer can survive a restart of the computer hosting the RFT service. RFT will continue to attempt to finish all transfers until a user specified deadline.

The significance of RFT to the GridFTP Overlay Network service is that RFT provides a higher level service than the overlay network. Ideally RFT should be able to make use of the proxies with little or no modification. RFT only accepts URL pairs as input which means in order make use of both RFT and proxies to split GridFTP connections, the proxy information would have to be embedded in URLs. The GridFTP servers would then interpret the requests appropriately.

2.2 Network Protocols

This section discusses the network protocols commonly used by most applications in todays Internet, including file transfer applications.

Network protocols are typically defined in layers with each layer responsible for different functions [81]. Different protocols at the same layer tend to provide a

different quality of service such as whether packet delivery is guaranteed or not. The International Standards Organization (ISO) defined a seven layer Open System Interconnect (OSI) model as a networking standard [45]. In practice, a simplified four layer model was actually put into wide use. It is referred to as the TCP/IP protocol suite or the Internet protocol suite [81]. The suite's four layers include :

- Application Layer: Application specific protocols such as FTP or telnet.

- Transport Layer : Provides the movement of data between two hosts. The two most widely used transport layer protocols are the Transmission Control Protocol (TCP) [71] and the User Datagram Protocol (UDP) [70]. UDP is a best effort protocol with no congestion control. In the absence of any application layer congestion detection, dropping packets at network interfaces and routers is the only way to control UDP traffic. Applications using UDP have to handle reliability themselves. TCP, on the other hand, provides reliable transmission with congestion control. This means TCP reduces the software development effort as it frees application programmers from implementing functionality related to transmission of data across the network. For these reasons, TCP is by far the most widely used protocol on the Internet and represents between 60% and 90% of the traffic [32]. TCP is discussed in detail in section 2.3.

- Network or Internet Layer : Provides functionality necessary for the movement and routing of packets around the network. The Internet Protocol (IP) [1] provides connectionless packet based, unreliable data transfer. In this case the term unreliable indicates that the IP protocol does not guarantee that packets will arrive, that the ordering of packets will be maintained or that the data is

not corrupted along the way.

- Link Layer : Handles all hardware level details of data transmission. This layer includes both the device driver in the operating system and the network interface card.

One important detail regarding the TCP/IP protocol suite is that the application layer is typically implemented in user space, while the other 3 layers are implemented in kernel space [81]. This is not a requirement of the TCP/IP protocol suite, but most TCP/IP protocol stacks have been implemented in this manner. The result of this convention is that a system designed to improve throughput that operates at the transport or network layer requires a modification of the kernel. These types of changes can be difficult to deploy. In comparison any modifications done are the application layer are much easier to deploy because they only require building the new application. The advantage of modifying a lower layer protocol such as the TCP at the transport layer means that all applications using the protocol can make use of the change with little or no modification.

2.3 Transport Control Protocol

TCP is one of the most widely used protocols on the Internet and is the most often used protocol for file transfer. However, TCP fails to utilize a significant portion of available bandwidth in some environments. This section describes the TCP protocol then explains some its shortcomings. The environments in which TCP fails to effectively utilize bandwidth and some of the reasons for this ineffectiveness are discussed.

2.3.1 The TCP Congestion Control Protocol

The TCP protocol [71, 5] provides guaranteed, in-order-delivery of packets with congestion control. This section describes the TCP congestion control protocol and the problems it encounters in modern networks. The TCP congestion control protocol is window based. The size of the window defines how much data can be in transit at any one time. The protocol consists of four algorithms; slowstart, congestion avoidance, fast retransmit, and fast recovery. These algorithms make use of the following variables:

- Flight Size : The amount of data that has been sent but not acknowledged.
- Round Trip Time (RTT) : The interval of time between the moment a packet is sent and the moment acknowledgment of the same packet is received. The RTT of a particular connection is assumed to vary little and an estimate of RTT is maintained for each connection.
- Maximum Segment Size (MSS) : The maximum segment size is the maximum size of a TCP/IP packet, excluding the TCP/IP headers.
- Receiver Window (*rwnd*): A TCP variable advertised by the receiver indicating the amount of data it is willing to receive. The flight size must never exceed rwnd. This prevents the sender from transmitting data faster than the receiver can consume it.
- Congestion Window (*cwnd*): A TCP variable used in determining the amount of data to send. Cwnd is increased and decreased in response to congestion (or lack thereof) on the network. The flight size must never exceed cwnd.

- Loss Window (LW): The size of cwnd after the sender detects loss using a retransmission timer indicating an acknowledgment for a packet has not been received after a delay based on RTT or a constant upper bound. It is the size of one full sized segment.

- Restart Window (RW): The size of cwnd after TCP restarts transmission after and idle period.

- Initial Window (IW): The initial size of cwnd.

- ssthresh : A variable used to determine whether to use the slow start or congestion avoidance algorithm.

- Timeout : The maximum duration of time allowed for a segment to be acknowledged. This is only used when RTT is unknown.

The TCP congestion control protocol begins with the slowstart algorithm. This phase is used to determine the available bandwidth over the link. During this phase cwnd is increased at most one MSS for each ACK received that acknowledges new data. The increase in window size is linear with respect to the number acknowledgments received but the rate at which acknowledgments are received increases exponentially because the full window size is sent in a fixed interval equal to the RTT. When slowstart begins, the window size will be one segment. When this segment is acknowledged in approximately on RTT, the window size is increased to two. When the acknowledgments are received for the two segments (again in one RTT) the window size will be increased to four. The window size is doubling every RTT.

This provides an exponential increase in cwnd. TCP stays in slowstart until ssthresh is reached or congestion is encountered.

The congestion avoidance algorithm is used when cwnd is greater than ssthresh. During this phase, the congestion window is increased by at most MSS plus the size of TCP/IP headers per RTT. This linear response is referred to as additive increase.

Congestion is detected by either a timeout or duplicate acknowledgments. Congestion is detected by a timeout when the sender has received no acknowledgments from the receiver for specified amount of time after data has been sent. The receiver can also actively indicate a dropped packet by sending an acknowledgment when it receives out of order packets. These are referred to as duplicate acknowledgments because they indicate the same last data packet received (IE. the last one received in-order). When congestion is encountered, ssthresh is cut in half (subject to a minimum of 2 MSS). This response is referred to as multiplicative decrease. The additive increase in cwnd during the congestion avoidance phase and multiplicative decrease when congestion is encountered, just described, is often referred to as TCP's Additive Increase/ Multiplicative Decrease (AIMD) algorithm.

There are some exceptions to the Multiplicative decrease. If congestion is indicated by a timeout, the congestion window is reduced to the Loss Window size. If the congestion is indicated by 3 duplicate acknowledgments, ssthresh is set to 1/2 the current flight size. The congestion window is set to (ssthresh + 3) $\times$ MSS. This keeps the number of packets in flight equal to the new ssthresh because of the 3 duplicate acknowledgments implies 3 packets have left the network. Each new acknowledgment indicates that another packet is no longer in flight and a new packet is sent. This process is maintained until new data has been acknowledged. This is

referred to as the fast retransmit/fast recovery algorithm. In addition to detecting congestion, the congestion window is also reduced to Initial Window if no data has been transmitted for more than than one retransmission timeout.

2.3.2 TCP Metrics

There are a large number of metrics that can be used to evaluate the performance of the TCP protocol; the protocol used by GridFTP. This thesis focuses on the following metrics with the following two definitions:

- Bulk Transfer Capacity : Bulk Transfer Capacity (BTC) is a measure of a set of links' ability to transfer large amounts of data over a single congestion aware transport layer connection such as TCP [58]. It is more formally defined as:

  $BTC = unique\_data\_transferred/elapsed\_time$

  BTC is a measure of achievable effective throughput, as opposed to available bandwidth. Available bandwidth is the maximum rate of a new flow that will not reduce the rate of existing flows [49]. Achievable effective throughput is the unique data throughput actually achievable by an application. This is affected by a variety of factors including characteristics of the link such as latency, maximum transmission unit, packet loss and the available bandwidth. BTC is also affected by the specific implementation of TCP being used. TCP's congestion control algorithm, as formally defined by RFC 2581 [5] leaves enough choice in implementation of the algorithm to have a significant effect on throughput [58]. These choices include the specific way in which cwnd is increased during congestion avoidance. Whether or not specific loss recovery algorithms,

such as selective acknowledgments (SACK) [59], are used can also have a significant effect on throughput. SACK is an optional extension to TCP that allows the receiver to acknowledge packets that have arrived before earlier sequenced packets.

Achievable throughput is usually lower than available bandwidth but it has been noted [49] that achievable bandwidth may be greater than available bandwidth due to the effects of the new stream on pre-existing streams. New streams may cause pre-existing streams to reduce their window sizes, opening up additional bandwidth on the bottleneck link. In addition to network characteristics, throughput can also be affected by disk access speeds and the load on the computer. The BTC of a connection is the best predictor of the goodput of a specific data transfer. Goodput uses the same formula given above for BTC and is a common measure of the performance of an individual transfer.

- Variance in throughput: Variance in throughput is the degree to which the transfer time of a file is different depending on when it is sent. Some of this variance is the result of non-stationarity in network traffic levels. Some of this variance is the result of the protocol used. A lower variance in transfer time is useful to meta-schedulers that must co-allocate network, disk and compute resources.

2.3.3 Models of TCP performance

There is a great deal of literature on modelling TCP/IP. In [60] a reasonably simple model is used that predicts throughput quite well when the probability of losing

packets is small. Given a connection's maximum segment size (MSS), round trip time (RTT) and loss probability p, the model of bandwidth (BW) can be stated as :

$$BW = \frac{MSS * C}{RTT\sqrt{p}} \tag{2.1}$$

In this model, C is a constant that encapsulates several details about the specific TCP implementation and the loss model used. For example given periodic loss (as opposed to random), and a TCP implementation that acknowledges every packet, $C = \sqrt{3/2}$. With a delayed acknowledgment strategy, $C = \sqrt{3/4}$. This model assumes that the connection is limited by the congestion window and not the receive window. It also does not model timeouts or slowstart.

A more complicated and more accurate model is presented in [66]. This model takes into account timeouts. Taking timeouts into account may have relevance in understanding the throughput of split-TCP connections as it has been suggested that the lower variance of RTT may result in fewer spurious timeouts [84]. A spurious timeout means it was determined that a timeout occurred when the data was not actually lost, there was just significant delay. This makes the TCP connection needlessly throttle back throughput. The extent to which this effects the throughput has never been examined.

The model of TCP performance given in equation 2.1 provides insight into why splitting a TCP connection can improve throughput. In a split connection with two segments, the two segments have a predicted throughputs BW_1 and BW_2 of:

$$BW_1 = \frac{MSS_1 * C}{RTT_1\sqrt{p_1}},\ BW_2 = \frac{MSS_2 * C}{RTT_2\sqrt{p_2}}$$

Where:

$C = \sqrt{3/2}$

While the original connection has a throughput BW_t of :

$$BW_t = \frac{\min(MSS_1, MSS_2) * C}{(RTT_1 + RTT_2)\sqrt{p_1 + p_2 - p_1 p_2}}$$

BW_t is always less than BW_1 and BW_2, therefore, a throughput gain for a split connection is guaranteed according to these models if the assumptions of the models hold, and the throughput of the split connection can be assumed to be min (BW_1, BW_2).

2.3.4 TCP on High Bandwidth Delay Product Links

The TCP congestion protocol has very poor link utilization in high bandwidth, high latency networks. These have also been referred to as high bandwidth delay product (HBDP) networks or long fat networks (LFNs) [48]. The utilization is particularly low when individual flows have HBPDs [52]. The bandwidth delay product (BDP) is the amount of unacknowledged data that must be on a link in order for the available (or maximum) bandwidth to be fully utilized. It is defined as the product of the bandwidth of the slowest link along the path and the round trip time. A *high* bandwidth delay product link is one in which the BDP is greater than the buffer space available in the routers along the path [55].

The poor performance of TCP on HBPD links is in part a result of its oscillation around the ideal send rate. TCP must do this because of the binary nature of the congestion signal; either the network is congested and a packet is dropped or it was not congested. When packet loss occurs, it was assumed to be because a buffer in

the network was full and a packet was dropped. TCP cuts its throughput in half when this happens. It was assumed that this would not waste bandwidth because the bottleneck link had a full buffer to empty out [47]. This leads to the rule of thumb that the router buffer be as large as the BDP [90]. However, this requirement is not particularly feasible with HBPD networks with individual flows in the range of 1 - 10 Gbps since the buffers required would be too expensive [52]. While using very large buffers might help overcome the oscillation of throughput in TCP flows, it would also introduce undesirably large amounts of jitter and queueing delay. In addition, large buffers tend to result in a synchronization of flows [72]. A synchronization of flows can result in underutilization of the network because all the TCP flows will experience loss and reduce their window size simultaneously. The simultaneous reduction in window sizes may cause the aggregate window size of all flows to be too small to effectively utilize the network [78].

In addition to requiring large buffers in routers for effective utilization of bandwidth, TCP requires very low packet loss rates in order to maintain throughput on the order of Gbps and higher. TCP's AIMD response maintaining large cwnd requires low loss probability due to the saw tooth pattern of the TCP congestion avoidance algorithm [27, 52]. An average congestion window for a standard TCP implementation is $1.2/\sqrt{p}$ segments, where p is the loss probability of a packet [31]. This response function implies standard TCP may fail to perform effectively in HBPD networks even with well tuned TCP implementations. The following example will demonstrate the difficulty.

Given a standard segment size of 1500 bytes, a round trip time (RTT) of 0.1 seconds and available bandwidth of 1 Gbps the following analysis can be made. The

bandwidth delay product of this network is 100 Mb or 12.5 MB. In order to fully utilize the link the window size would have to be 12.5 MB. This is 12.5 MB/1500 Bytes = 8738 segments. Given the response functionm, this implies a loss probability of $(1.2/8738)^2$ or approximately 1.8×10^{-8}. A packet loss probability of 10^{-8} represents a bit error rate of 10^{-12} [52] assuming no packets are lost due to congestion.

TCP also suffers from low link utilization because the response to newly available bandwidth is too slow. The linear increase of 1 packet is too slow when additional bandwidth becomes available [52] such as when a TCP stream leaves. On HBDP networks, realizing the additional bandwidth can take on the order of minutes. It is also not feasible to generally increase the window at a faster rate because when the window is small, even increasing by 1 packet per RTT is too aggressive to maintain stability [47, 72].

TCP is also well known to be unfair to flows with a relatively high RTT [55, 40]. This is primarily the result of the fact that TCP increases its window size at a rate tied to RTT in both slowstart and congestion avoidance. This results in throughput being inversely proportional to RTT when multiple connections share a bottleneck link [55].

If for any reason, the TCP window is not as large as the BDP the link will be under-utilized. This factor indicates the importance of properly sized TCP buffer sizes. This requirement results in a large number of computers poorly configured to perform well over HBPD links, including high performance computers intended for scientific research [74].

2.3.5 TCP in Heterogeneous Networks

TCP interprets all packet loss as a result of congestion only. However, many networks encounter relatively large amounts of uncorrectable physical bit errors. Poor utilization as a result of this erroneous assumption has been observed in wireless networks [8], satellite links [12] and coaxial cable networks [20]. TCP treats these packet losses as a congestion signal and reduces the window size. This results in the window size never getting large enough to fully utilize the network.

TCP has also been shown to perform poorly over asymmetric links; links in which the characteristics of the path taken by the acknowledgements (rather than the data) affect throughput. Limited bandwidth, high loss or a high variance in RTT on the acknowledgements path can result in asymmetry. Asymmetry is often associated with media such as cable modem networks, satellite [24] and Asymmetric Digital Subscriber Line (ADSL) networks [9]. Asymmetry can, however, occur over traditional wired links as well. This can occur due to a faulty line in one direction or the routing in each direction may be different. Analysis of routing asymmetry has previously indicated that 49 % of internet paths show some assymetry and 20% show differences in more than 1 hop [68].

If the bandwidth on the path of the acknowledgements is not large enough for the acknowledgements sent, this will clearly limit throughput. Whether the difference in bandwidth has an effect on throughput depends on the normalized bandwidth ratio [9]. The normalized bandwidth ratio, k, is defined as the ratio of the raw bandwidths divided by the ratio of the packet sizes. If more than one acknowledgment is sent for every k data packets, the asymmetry will affect throughput. For

example, if 1500 byte data packets have available bandwidth of 10 Mbps and the 40 byte acknowledgements have an available bandwidth of 100 Kbps, the normalized bandwidth ratio is 2.67. If the TCP implementation acknowledges every 2nd packet, the connection will be limited by the asymmetry of the link. The effect of bandwidth asymmetry is significant for split-TCP implementations because, unless split TCP is done at a router, the second connection can create bandwidth asymmetry for the first by using up the bandwidth on the outgoing link from the proxy. The outgoing data channel interferes with the acknowledgements for the incoming data channel. Depending on the specific topology, asymmetric bandwidths can cause burstiness in the sender, slow rates of window growth, idle periods, and a disruption of fast retransmit. A high variance in RTT means the TCP connection cannot determine a timeout in a reasonable amount of time. This can lead to large periods of time with no throughput at all when the variance of the acknowledgment path is very high [9].

2.4 Overlay and Peer to Peer Networks

Closely related to the grid computing are peer-to-peer (P2P) technologies and overlay networks. Peer-to-peer systems are distributed systems consisting of interconnected nodes for the purpose of sharing resources [7]. There is generally a large focus on their ability to function without a central server and have transient populations of hosts [7]. In most cases this has been restricted to utilizing a computer's bandwidth and storage abilities to distribute files. One of the significant differences between P2P technologies and Grid technologies is grid technologies have a stronger focus on standardization to realize computing resources as a utility, or an infrastructure.

The peer-to-peer domain on the other hand has tended to consist of stand-alone applications that do not inter-operate [7].

Overlay networks and P2P systems are closely related concepts. Overlay networks create a virtual topology on top of an existing network or physical topology [22]. Most peer-to-peer systems are overlay networks because they create a virtual topology over the Internet to communicate or distribute data. Overlay networks tend to be somewhat self organizing so they can also be classed as peer-to-peer networks. Whether a system is referred to as an overlay network or a peer-to-peer network seems to be a function of the system's properties. If most of the benefits of the system are a function of the fact that it has its own virtual topology, it is referred to an overlay network. If the systems benefits are primarily from the distributed use of peers resources, it is referred to as a peer-to-peer system. Overlay networks have been used to improve the throughput and connectivity of the Internet [6] and are discussed more thoroughly in Chapter 3.

2.5 Summary

This chapter presented background on the technologies relevant to the GridFTP Overlay Network Service. These technologies provide the security, resource discovery, data management and data transfer services required to enable the use of proxies to split a GridFTP connection into a series of shorter contiguous TCP segments. TCP has poor utilization on high bandwidth delay product links and in heterogeneous environments. The splitting of a connection into several shorter segments can improve the performance of TCP in these environments. The GridFTP Overlay net-

work could be considered a peer-to-peer technology because it is distributing the use of computer memory, P2P technologies are, however, typically more resilient to transient populations and self organizing than the GridFTP Overlay Network Service is designed to be.

Chapter 3

Related Work

There is a large body of research related to high performance data transfer. Because TCP is the most commonly used protocol for data transfer, there is also a great deal of work done on improving TCP performance on links with high bandwidth delay products (HBDP), high error rates, and heterogeneous networks. This research has included the creation of many variations of TCP, application level solutions, and new transport layer protocols. Due to the large volume of work in these areas, a complete survey is not possible here. This chapter outlines some of the work on improving TCP or alternative protocols for high performance data transfer.

3.1 Variations of TCP

TCP was originally specified in RFC 793. It has since been refined, updated and extended with RFCs 2581 and 3390. The TCP algorithms described in section 2.3 were first completely implemented in TCP Reno. A great deal of research has been done suggesting further improvements to TCP. Some changes represented fundamental improvements such as Selective Acknowledgments (SACK) which was widely adopted. A number of modifications to TCPs congestion control algorithm have been suggested to overcome TCPs utilization issues in high bandwidth delay product (HBPD) environments. Some of these changes were fundamentally different than the congestion control algorithm suggested in RFC 2581 such as using changes in

round trip time (RTT) as a congestion signal whereas other simply made the existing additive increase, multiplicative decrease (AIMD) algorithm more aggressive. This section describes a small portion of these suggested changes. A full survey is not possible here due to the volume of research in this area.

3.1.1 SACK

TCP SACK [59] allows the receiver to acknowledge packets that have arrived before earlier sequenced packets. This allows the sender to resend just the dropped packets as well as new data. Without SACK, the sender must either retransmit at most one dropped packet per RTT or retransmit successfully received packets [29]. TCP SACK was originally designed to improve performance on heavily congested networks resulting in multiple packet drops. However, it has also been shown to improve performance on uncongested high-speed wide area networks when errors are clustered together [65]. As of 2004, studies have indicated that 68% of web servers indicated they were SACK enabled by sending the SACK_PERMITTED option although only 54% actually made use of the SACK options [62].

3.1.2 VEGAS

TCP Vegas [16] differs dramatically from other TCP implementations in that it does not rely exclusively on dropped packets to detect congestion. Other TCP implementations are dependent on actually causing losses due to congestion to discover the available bandwidth.

As a TCP connection's send rate approaches available bandwidth, the packets for the stream will begin to fill the buffers at the bottleneck link and the sending

rate will become lower. At the same time, as the window size increases, the sending rate is expected to increase. Once the sending rate or throughput reaches available bandwidth, the increase in window size will only fill the buffers at the bottleneck link and not increase actual throughput. TCP Vegas measures the actual sending and compares it to the expected send rate. Vegas uses the difference between actual throughput and expected throughput to maintain enough packets in the buffers to respond to increases in available throughput while still avoiding congestion losses. It uses two parameters, α and β, to decide when to increase or decrease throughput. So long as the difference, *diff*, between the expected and actual send rate is between the parameters α and β, Vegas does not modify the window. If $diff < \alpha$ the window will be linearly increase while if $diff > \beta$ the window is linearly decreased.

The change in RTT is also used as a signal to change from slowstart to congestion avoidance. In addition to using changes in delay to throttle the send rate, TCP Vegas uses the receipt of some ACKS as a hint to check for timeouts. This results in significantly fewer timeouts [16].

The congestion avoidance mechanisms in TCP Vegas result in very consistent throughput and the familiar sawtooth pattern of instantaneous throughput is gone. This results in significantly better utilization of the link as well as considerably fewer retransmissions [16].

TCP Vegas has been shown to have significantly better network utilization over HBPD links [30, 50, 24] because it does not rely exclusively on packet drops as a congestion signal to discover available bandwidth. This avoids losses and the significant time required to recover from them over high RTT links. Vegas has the added benefit of not being biased against flows with a long RTT [63].

Although Vegas avoids the performance issues associated with the time required to recover from loss, it can often enter the congestion avoidance phase too early, resulting in a long (on the order of minutes) phase of underutilization [85]. Vegas is rarely implemented on production systems. This may be the result of a demonstrated poor compatibility with TCP Reno [63].

3.1.3 HighSpeed TCP

HighSpeed TCP [31] proposed to make modifications to TCP's congestion control mechanism for use with TCP connections with large congestion windows. The modified behavior would not take place in heavily congested environments which would tend to keep window sizes small.

HighSpeed TCP takes three parameters `Low_Window`, `High_Window`, and `High_P`. When the Congestion window is less than `Low_Window`, the response to congestion is the same as the TCP congestion control algorithm as defined in RFC 2581. When the congestion window is larger than `Low_Window`, HighSpeed TCP responds more aggressively. The response becomes a function of the current window size determined via a lookup table. The function uses the variables `High_Window` and `High_P`. These increases and decreases in the congestion window become similar to the aggregate congestion window of N parallel TCP streams where N grows as a function of the size of the congestion window [31].

HighSpeed TCP does perform well in a HBDP environment, however TCP Vegas still outperforms it [50]. The more aggressive increase and smaller decrease in window size allows HighSpeed TCP to tolerate larger loss probabilities in HBPD environments [19] but it also seems to make the RTT fairness problems that TCP

experiences even worse [50].

3.1.4 FAST

Fast Active-Queue-Management Scalable TCP (FAST) [19] was developed for HBDP networks. Its congestion control algorithm uses queuing delay, in addition to loss detection as a congestion signal. The use of queuing delay in controlling the window size to maintain a constant number of packets in the buffers makes the algorithm similar to TCP Vegas [19].

FAST TCP was found to outperform TCP Reno as well other high performance TCP implementations such as HSTCP [31] on several metrics including throughput, fairness, responsiveness and stability [19]. Deployment issues cannot be evaluated as inter-protocol fairness and performance have not been published. Because FAST is similar to Vegas, it may suffer the same compatibility issues with Reno [63]. FAST has not been compared to Vegas in side by side experiments.

3.2 Active Queue Management

Active Queue Management (AQM) involves the routers in the management of congestion in order to more efficiently manage packet drops and signal to endpoints that congestion is occurring. In the absence of AQM, packets are simply dropped when there is no buffer space available. This is referred to as a "Tail Drop" queue management scheme. Random Early Detection (RED) [14] queue management uses two queue length thresholds, minth and maxth to determine when to drop packets. When the average queue length is at less than minth, packets are not dropped. When

the average queue length is between minth and maxth the router drops packets with probability p, where p is determined by the average queue length and grows at the average queue length approaches maxth. When the average queue length is above maxth, packets are always dropped.

Explicit Congestion Notification (ECN) [73] gives the router the ability to signal congestion by setting a CONGESTION_EXPERIENCED (CE) bit in the IP header for ECN enabled IP and transport protocols. The receiver then relays this information to the sender. The transport protocol is expected to treat a packet with the CE bit enabled as a packet loss. This places the responsibility of managing how the CE signal is used onto the router. It is suggested for instance, that the CE signal should be based on some sort of average rather instantaneous queue size [73]. Performance results for RED used in conjunction with ECN indicate it does improve performance for bulk transfers; it has the effect of avoiding timeouts, which has the potential to improve performance dramatically [76].

3.3 Application Level Solutions

There are a large number of application level solutions that were designed to overcome TCPs poor performance in HBPD environments. These solutions often take the form of application level protocols that make some use of TCP underneath. Some solutions involved opening multiple TCP streams, while others use UDP for data transfer and a TCP connection to provide rate control. One of the big advantages of application level solutions is their easy deployment. This section provides an overview of some of these solutions.

3.3.1 Striped or Parallel TCP

MulTCP [21] was developed to provide differentiated service levels at the transport level. MulTCP modified the cwnd and ssthresh to simulate the response of N separate TCP connections. One difference from the simulation of N streams is that it begins by sending 3 packets for each ACK received (instead of N). MulTCP was not implemented to improve performance in HBPD environments but it was recognized that MulTCP could be useful in those environments [36] because it makes TCP more aggressive.

Since MulTCP was developed, several several file transfer applications and libraries have made use of parallel or striped TCP. XFTP [4] and GridFTP [2] are both extensions of FTP that implement striped TCP at the application layer by creating multiple data channel connections. Psockets [80] is an application level socket library with an API similar to the standard socket libraries. The library handles opening and closing of sockets and reassembly of striped data.

A large portion of the benefit of striped TCP comes from changing the response to losses of TCP. Several TCP streams in aggregate will more aggressively increase their aggregate window size and less aggressively decrease there aggregate window size. This allows the aggregate TCP stream to achieve a higher throughput or average window size with the same loss rates. The end result is that multiple TCP connections may receive a higher throughput than a single TCP connection with large TCP buffer sizes [4].

An application that makes use of striped TCP may receive greater performance at the expense of other connections. This is because TCP is designed to converge on

a fair allocation of bandwidth. If an application uses two TCP connections instead of one, it is allocating itself twice the appropriate bandwidth.

Using parallel TCP streams also increases the effective TCP buffers sizes of the transfer. In practice, it is a very lengthy process to tune TCP parameters at a host (and it needs to be done at both ends) and it is often not done. This tends to leave the buffers at the default of 64KB, far smaller than the bandwidth delay product on many connections in scientific computing environments. As a result, striping TCP has also been suggested as a way to overcome poorly configured computers [80].

One thing that has likely restricted the adoption of striping is that an application or striping implementation needs to know how many TCP stripes to use. It would be easy to continue increasing the number of streams until bandwidth was maximized but it is not clear how to distinguish between effectively using the network and taking bandwidth from others. Models to determine the number of parallel streams have been developed [46] that utilize bottleneck link bandwidth, RTT and TCP socket buffer size. This information could conceivably be obtained from in flight transfers to tune the transfers as they happen. The utility of this and its implications on fairness have not been addressed.

Striped TCP is useful in environments where TCP's assumption that loss is due to congestion does not hold, or when differentiated service is the goal. It can also be used to overcome poorly configured computers and has uses over high performance networks. Performance is also improved in HBPD networks because the effective AIMD algorithm is more aggressive.

3.3.2 SABUL

The Simple Available Bandwidth Utilization Library (SABUL) [38], is an application level protocol designed for high performance data transfer on HBDP networks. SABUL uses UDP for a data channel and TCP for a control channel.

SABUL adjusts its send rate at a constant interval as opposed to one dependent on RTT in TCP. The interval for rate adjustment is 0.1 seconds. This was chosen by trial and error as an acceptable trade off between efficiency and fairness. At these fixed intervals, if the exponential moving average of the loss rate is less than a loss rate threshold, the inter-packet transmission time is incremented by a constant amount that scales to the current rate. The inter-packet transmission is reduced by a factor of 1/8 when the receiver notifies the sender of a loss. Any out of order packet is considered a loss. This amounts to a modified additive increase, multiplicative decrease (AIMD).

In addition to this rate based control, sending data is stopped when the number of acknowledgments exceeds the maximum flow window size (which is determined by the application). This is an onerous requirement to put upon applications as the authors suggest it be set to a product based on the bandwidth. Bandwidth is often difficult to determine because there is no standard protocol for routers or switches to indicate their maximum bandwidths to end-points.

In order for SABUL to work effectively, the loss rate threshold must be tuned to be higher than the physical bit error rate. This parameter appears to be a constant however and is not negotiated. A network topology in which there is a leg with a high physical bit error rate (eg. wireless) would require this to be higher than is otherwise

optimal. For this reason, it is unlikely SABUL would perform well in environments with high physical bit error rates or heterogeneous network environments in which some links have high physical bit error rates.

SABUL is very effective at utilizing high bandwidth, high delay links and in general achieves over 90% bandwidth utilization on Gigabit wired networks. SABUL also meets intra-protocol fairness as well as being reasonably TCP friendly. SABUL has not been tested in high loss networks and due to its design, it may not be very effective in these environments. It also currently puts a heavy requirement on the application to properly set the maximum flow rate. For this reason, it is very effective for the high bandwidth environments it was designed for, but may suffer in more heterogeneous environments.

3.3.3 RBUDP

Reliable Blast UDP (RBUDP) [41] is another application level protocol that uses UDP and a TCP control socket for retransmitting packets. RBUDP is a very aggressive bulk data transfer protocol. It is intended only for very high bandwidth, dedicated or QOS enabled networks, such as networks with dedicate light path reservations.

The application provides a send rate to the protocol. RBUDP then sends all the data via UDP at the specified send rate. Once data transmission is complete, a signal is sent to the receiver. The receiver then responds with the packets it is missing, which are then sent by the sender as the specified send rate. This process is repeated until the receiver has received all the packets. Knowing the appropriate send rate is critical because the protocol has no in transit rate discovery/adaptation mecha-

nism. If the send rate is overestimated, overhead due to retransmission becomes very large [92].

Experimental results have shown that RBUDP can be very effective at utilizing bandwidth when the appropriate send rate is known. This can be difficult as it entails not only knowing the available bandwidth, but the capacity of the receiver to receive the data. For example, during their experiments, the authors noticed a 33% loss rate, despite using a send rate of 5% less than available bandwidth. The recieving computer was not capable of moving the data from kernel space to application memory fast enough.

RBUDP may be effective in the environment is was designed for, when the users have a guaranteed QOS and the applications can discover the appropriate send rate. Fairness and adaptation to dynamic available bandwidth are not one of the goals for the protocol, making it inappropriate for traditional IP, shared networks.

3.4 New Protocols

New protocols have been developed for high performance data transfer that often perform very well in many environments in which the performance of TCP is poor. They are not often deployed however as the cost can be prohibitive. They require deploying a new protocol stack and sometimes a change to routers. There are a great deal of obstacles to achieve these kinds of changes.

3.4.1 XCP

The eXplicit Control Protocol (XCP) [53] builds on Explicit Congestion Notification (ECN)[73] by adding explicit reductions and increases in bandwidth at the router. The general scheme of the control protocol is still AIMD, but is essentially implemented at the router and XCP need not be limited to these. The decoupling of efficiency control from fairness control is one of the main contributions of this work and results in significant benefits. In TCP, the same rule (AIMD) is used to achieve both fairness and efficiency. In both TCP and XCP fairness is defined as if N streams are in contention for a bottleneck link each stream will converge on a send rate equal to 1/Nth of the available bandwidth. .XCP has separate controllers to achieve this; an efficiency controller and a fairness controller. The controllers provide feedback to the source using the congestion headers.

The efficiency controller computes the desired increase or decrease in aggregate flows (ϕ) for a particular link at control intervals. Control intervals are equal to the average RTT. The fairness controller (FC) then distributes the positive or negative ϕ among the flows. The FC enforces AIMD. If ϕ is positive, it distributes the increase equally among the flows, if ϕ is negative, the decrease is distributed proportionally to the current cwnd. Convergence to a fair distribution may stall when $\phi \approx 0$, so the FC also ensures that at least 10% of the traffic is redistributed each control interval. The necessary changes in cwnd are relayed back to the sender in a congestion header in the acknowledgments.

XCP outperforms TCP in both conventional and high bandwidth environments. It achieves fair bandwidth allocation, high utilization, small standing queue size, and

almost no packet loss. XCP can also be made TCP friendly. The main drawback of XCP is that it requires router support. This support is not onerous as there is no per flow state maintained by the router, but it is a very large obstacle to deployment. This deployment may still be reasonable on high-performance scientific networks [28]. These networks are more expensive and the extra expense in the routers can more easily be justified by the increased utilization.

3.5 Split-TCP

The under utilization of the bandwidth in long fat networks (LFNs) and heterogeneous networks has led to the suggestion of splitting a TCP connection into several TCP segments connected in series, usually with data buffered in between segments. Split TCP connections are one type of *Performance enhancing proxy* (PEP) [13]. PEPs are intended to mitigate performance degradation due to the characteristics of the link. For this reason, split-TCP implementations were first suggested for heterogeneous environments. Previous work in the area of split-TCP consists of analytical models, applications to specific physical network topologies, and, more recently, generalized split-TCP applications.

In [25], the authors present an analysis of split TCP connections. They identify several factors affecting the performance of a split TCP connections such as the asymmetry of the links. While they do suggest that the best way to split a TCP connection is such that the performance of the two links are similar, they do not present a general algorithm as to when it is useful given a set of link characteristics. Their analysis also assumes identical Maximum Segment Sizes (MSS) while several

analytical models have pointed out that the performance of a single TCP connection in steady state is proportional to the ratio of MSS to round trip time (RTT) [60].

An analytical model of split-TCP throughput incorporating packet loss, round trip time, and the receiver's advertised window is presented in [82]. This model also does not incorporate MSS. The analysis is of throughput (number of bytes transferred), not goodput (number of unique bytes successfully transferred). The work analyzes throughput of the split connection as a function only of the 2nd connection. However, as pointed out in [25], this is only valid if he throughput of the second connection is not starved by the first.

I-TCP [8], *Snoop* [10],and *M-TCP* protocol [17] all use proxies to enhance performance in a wireless environment. The losses in a wireless environment are often not due to congestion and the sender unnecessarily reduces the congestion window size. The proxy maintains a buffer of its own and hides the losses from the sender so the congestion window stays open. The protocols differ in their implementation. For example, M-TCP uses two connections; a normal TCP connection to the base station and a specialized M-TCP connection from the baystation to the wireless receiver. M-TCP does not violate end-to-end semantics because it only acknowledges a packet to the sender once it has received one from the wireless receiver. I-TCP uses two separate TCP connections which may be seen as violation of TCP's end-to-end semantics. Snoop operates at the link layer, maintaining its own cache to retransmit from and suppresses duplicate acknowledgments. Because Snoop does not send its own acknowledgments, it also does not violate TCP semantics. [20] examines the use of proxies to enhance TCP performance of hybrid fibre Coaxial (HFC)networks. Proxy servers are used to handle access from coaxial network. A proxy system to

improve performance over satellite links by using two proxies (one on each side of the satellite link) is proposed in [12]. This effectively splits the end-to-end connection into 3 separate links. In all cases, significant improvements in performance are observed as a result of decoupling the higher bit error wireless or Coaxial links have from the high overall RTT.

Splitting a TCP connection improves performance in several related ways. The most significant reason for improved throughput is that TCP uses RTT as a clocking mechanism to pace increases in the window size. By splitting a connection into two or more smaller ones, each individual connection will more aggressively increase the window size during both slow start and congestion control when compared to a single TCP connection. This results in TCP attaining steady state earlier, and recovering from errors more quickly. The rapid recovery from errors improves the performance of TCP in many instances in which TCP has poor performance. For example, TCP assumes that all packet drops are due to congestion. When this assumption is wrong, as in wireless networks, the faster recovery time dramatically improves performance [8]. Even in the absence of any unusual network conditions, the lower RTT should improve steady state performance as demonstrated from equation 2.1.

In addition to the benefits gained from lowering the RTT, split-TCP connections also obtain other significant benefits. In the case of the HFC networks, further optimizations of TCP are possible because of guarantees such as in-order delivery of packets over one of the links [20]. The locality of retransmission has been pointed to as a reason for improvement [84] but this would seem to have little effect on most networks given that less than 1% of packets are typically lost [47]. Split-TCP also

benefits from reducing RTT variance because the TCP segments are shorter which allows TCP to make more accurate RTT estimates and avoid spurious timeouts [84]. Some of the most significant improvements have been found to come from simply routing around network pathologies such as assymetry [84] or poorly configured computers [74].

3.5.1 Generalized split-TCP implementations

The increasing complexity of the Internet has resulted in more situations in which TCP under-utilizes the network. This has resulted in an interest in generalized network split-TCP proposals [39, 57, 84, 75]. Generalized split-TCP proposals can be distinguished by not being designed for use on a specific physical topology, but for the general Internet. Decisions on where to split connections is based on some limited knowledge of the network, such as RTTs between hosts [57, 39], active probing [84] or passive probing [75].

Generalized split-TCP overlay networks [39, 57] have been proposed and tested primarily for the purposes of Web traffic. Overlay TCP [39] uses RTT exclusively while the strategy described in [57] uses both RTT and packet loss. These proposals tend to suggest using one or two fundamental characteristics of the network such as RTT and loss rather than actual previous file transfer performance. Neither of these papers addressed the accuracy required for the information used to determine where to split connections. This is important when using fundamental characteristics because adding a TCP stream to the network can increase the RTT and loss, skewing performance predictions by orders of magnitude [42].

The Logistical Session Layer (LSL) [84] utilizes a split-TCP connection by imple-

menting an application level socket library. Although the authors claim LSL operates at the session layer, it is in fact a transport layer service. It is the role of the transport layer to achieve cost effective transfer of data, and the ISO model requires a one to one mapping between session connections and transport connections [45]. An LSL connection has many transport (TCP) connections.

LSL does not require kernel modification as both the socket library and the proxies at the end of the TCP segments run in user space. This greatly reduces the security concerns and makes deployment much easier. LSL utilizes network weather service (NWS)[91] to arrive at "good" places to split a TCP connection. This can be seen as a limitation as NWS utilizes active probes and has been found to be inaccurate on high bandwidth networks [86, 87].

The authors of LSL have restricted their study to differences based on RTT and they have not examined the effects of other network link characteristics. In addition, this work has restricted its study to small bandwidths and small files. One characteristic of many grid enabled sites is that they have considerable bandwidth, often of 1 Gbps or more. When bandwidth is this high, it results in other factors entering consideration such as the bandwidth being limited by the disk.

Generalized split-TCP implementations tend to implicitly assume that the bulk transfer throughput of a split-TCP connection will be the same as the throughput of a *normal TCP connection over the slowest link* [57, 84, 75]. While this assumption is useful to decide where to split a connection, the throughput of the split connection is rarely as good as the throughput of a normal TCP connection over the slowest link. There are several reasons for this.

One of the reasons for degradation of a split-TCP connection is the interaction

between the streams. The in-order semantics of TCP means that if a proxy node detects a packet loss it will not deliver any packets to the split proxy that come after the lost packet, even if they are in the proxy receive TCP buffer [57]. If congestion is experienced downstream, the proxies buffers may fill up. This will cause the proxy to send ACK packets with a receive window set to zero. This could conceivably limit the throughput of the downstream connection when it recovers and the buffers at the proxy are emptied out. Both of these examples point to a dependency between the two connections that may invalidate the assumption that the throughput will be that of the worst link. These effects can be minimized by increasing the TCP-buffer sizes. The two TCP streams may also interfere with one another on the wire. Specifically, the second TCP connection may interfere with the first connections ACKs. This possibility has received little study, possibly because most implementations assumed the proxy functioned as a router. This issue is explored in section 5.2.5

3.5.2 End-to-End Semantics

One argument often presented against splitting TCP connections is that it violates TCP's end-to-end semantics [57]. This may be important to maintain because for general application support because an application may assume this. It has in fact been pointed out this may not be strictly true [12]. The original specification for TCP [71] specifically states:

> *"An acknowledgment by TCP does not guarantee that the data has been delivered to the end user, but only that the receiving TCP has taken responsibility to do so."*

While it is unclear as to whether or not this permits a TCP receiver to delegate the responsibility of delivering a packet to another host, because the semantics of

TCP do not guarantee an application has received data, it is clear that a program cannot be confused by any violation of TCP semantics that do occur.

Regardless of the formal semantics for TCP, for a file transfer application the end-to-end acknowledgment within TCP is redundant. Generally, the file transfer would not be considered complete unless the file was written to disk error free. Because only the file transfer application can take steps to ensure the date was written to disk correctly, only the file transfer protocol needs to ensure that the data was actually received. This is in accordance with the end-to-end principal;

> *"The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end-points of the communication system. Therefore providing that questioned function as a feature of the communication system itself is not possible. Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement"* [77].

The end-to-end reliable data transmission semantics of a communication subsystem are generally useful for congestion control, and not useful to the application [77]. This is because most applications are not interested only if the data was received by the remote process but also whether the process responded to the data in some way. There is, therefore, no obvious harm in putting a performance enhancing proxy at the transport layer.

While it may be acceptable within TCP semantics to put a performance enhancing proxy (PEP) at the transport layer, there are reasons not to do so. TCP cannot make use of PEPs transparently because using PEPs implicitly assumes that bulk throughput is the priority of the application. This is not necessarily the case as other applications have other priorities. For example, interactive applications such as telnet or voice over IP are likely to put a priority on latency which would be

made worse by PEPs. This particular limitation on transport layer solutions can be overcome by making TCP context aware by passing information from the application. The network specific PEP solutions for the wireless domains is effectively a context aware implementation of TCP [89]. Putting generalized PEPs within the network would likely be an expensive proposition both in terms of the actual cost of the hardware and the management of access to the proxies. The use of PEPs could be limited to certain users, but it is difficult to properly manage buffer space usage and authorization among users at the transport layer. The Grid Security Infrastructure (GSI), on the other hand, can alleviate this concern. GSI provides mechanisms for fine grained access control to a user base that crosses multiple administrative domains. The addition of PEPs cannot be transparent, and the transport layer can not handle user based authentication properly. This suggests splitting a TCP connection should be done at the application layer, rather than transport layer.

3.6 Overlay Networks

Overlay networks have been used to improve TCP performance by changing the route of a connection such that the characteristics of the route taken are ones that TCP will perform better in. For example the new route could have a lower RTT or packet loss rate. Overlay networks often take the form of application level interfaces that are built on top of the network [6], transport [39] or application layer.

A Resilient Overlay Network (RON) is an application level overlay that exists on top of an existing network [6]. Nodes in a RON monitor the performance of the underlying network to detect outages and reroute through other nodes on the RON

when the direct path is no longer the best (or good enough). It was designed with 3 goals in mind: detection of network faults in less than 20 seconds, tighter integration with applications for routing decisions, and the abilty to express routing policy.

By rerouting packets through peer nodes, a RON can overcome network faults in 10s of seconds as opposed to several minutes [6]. The RONs tighter integration with applications allows a RON to optimize the route based on the application's specific needs. Routes can be optimized based on metrics such as latency, jitter or throughput. An expressive policy routing algorithm allows specific administrative issues to be addressed as well. For example, it can be specified that commercial traffic must not use the Internet2 educational backbone.

The RON architecture does not address scalability issues and the authors point out that a great deal of applications can still benefit from the target size of 2 to 50 RON nodes. While the main goal of RON appears to be addressing network outages, it also attempts to optimize routes based on several metrics, one of which is throughput.

RONs approach to improving throughput is to use formula based prediction using a modified version of the simple throughput model developed in [66]. This model assumes congestion is the limiting factor and represents an upper-bound on the throughput of the connection. Based on previous work examining the stability of available bandwidth [11, 69], RON only reroutes packets if the predicted improvement in upper bound is greater than 50%. RON managed to double TCP throughput in 5% of the samples. In 2% of the samples, RON increased throughput by more than a factor of 5. Meanwhile only 1% received a throughput of less than 50% of the direct path.

The RON architecture will capture any performance difference to be gained by rerouting alone. It will not capture the gains to be made by reducing RTTs and recovering from losses more locally that splitting the TCP connection will.

Overlay TCP (oTCP) [39] is an application level, overlay network designed to make use of the split-TCP in a manner transparent to the applications. Primarily targeted at web traffic, it observed improvements on 85% of the connections tested. The system relied on RTT exclusively to predict improvements. Although not clear from the poster paper, some the improvement is likely due to the connections being TCP buffer limited given that the client was *wget*.

3.7 Summary

This chapter outlined a small part of the large body of research on high performance data transfer. Because TCP is the most commonly used protocol for data transfer, a large portion of the work has been related to improving TCPs performance. Research on TCP has often focused on specific environments such as links with high bandwidth delay products (HBDP), high bit error rates, and heterogeneous networks because TCP is more prone to poor utilization in these environments. The fundamental reason TCP can perform poorly is that it uses a binary congestion signal. New transport layer protocols that make use of better congestion signaling are very promising but the deployment expense is a significant barrier to adoption. Other high performance data transfer solutions utilize TCP at the application layer because it minimizes deployment expense but tend to not be as widely applicable as TCP. Splitting a TCP connection has generally been used in specific environments and

trying to generalize it at the transport layer results in applicability issues because it prioritizes throughput over other characteristics such as latency.

Chapter 4

GridFTP Overlay Network

This chapter describes the GridFTP Overlay Network. The GridFTP overlay network is a system designed to enable split-TCP connections to improve file transfer performance. The system consists of a set of proxies at which to split GridFTP connections and the services required to effectively use such a system. The requirements and design rationale are first discussed followed by a top down description of the architecture. A detailed description of each component and the reasoning behind the approach taken are then discussed. Finally, a distinction between the problems of proxy placement and proxy selection are given.

4.1 Design Rational

The GridFTP Overlay network enables the use of split-TCP connections to improve GridFTP transfers. Using an existing protocol such as GridFTP allows the system to be used by existing clients. This means new client software does not need to be developed or deployed. More importantly, it also means services already designed to use the pre-existing protocols can make use of the service. The system is implemented at the application layer which means no modifications to the kernel need to be done. Modifications to the kernel are much more difficult to get deployed than application level software. This is the reason many high performance data transfer systems are implemented at the application layer. Finally a system designed to make use of

split-TCP on any network topology needs explicit information about the network to make routing decisions. The GridFTP overlay network presented in this thesis does not have dependencies with respect to a particular network information service such as Network Weather Service (NWS) [91] and makes use of simple passive network monitoring techniques. The prediction mechanism could, potentially use NWS to improve results [86] but it is not a requirement of the system. The GridFTP overlay network uses a standard interface to the network information with the ability to plug the best mechanism available to predict performance behind it.

4.2 Architecture

The GridFTP Overlay network consists of three primary components :

- split-DSI Module
- Split Choice Service (SCS)
- BTC Information Service

These components have dependencies on parts of the Globus Toolkit version 4 [33]. Predictions on the BTC of links from the BTC Information service is pushed into an MDS4 server. The SCS then pulls the BTC predictions to choose proxies for a specific transfer. The split-DSI Module requires the deployment of a GT4 GridFTP server. The Globus Toolkit and MDS4 are discussed in section 2.1. Figure 4.1 shows the software stack and dependencies of the GridFTP overlay network services.

Figure 4.2 depicts how a file is transferred using the GridFTP overlay network. The client developed to use the system first queries the SCS to determine a split

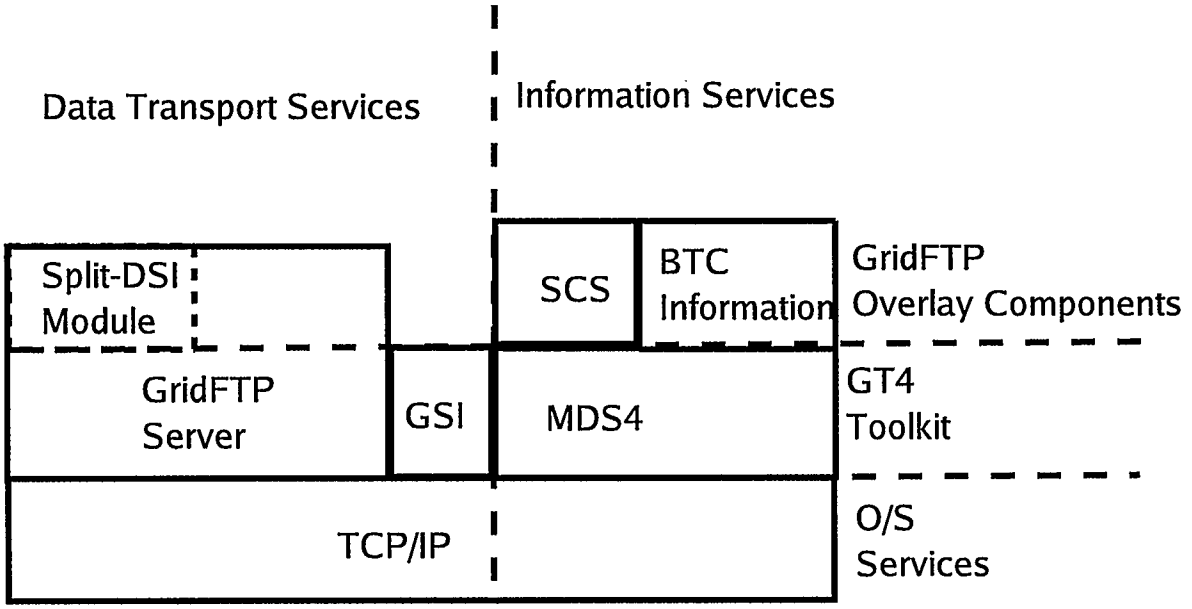

Figure 4.1: GridFTP Overlay Software Stack

connection giving a source A and a destination C. The SCS then queries an MDS4 server [79], an information service provider distributed with the Globus Toolkit (see section 2.1), for recent bulk transport capacity (BTC) predictions for all hosts A through E. This information is populated on an ongoing basis by a collector of results from normal GridFTP transfers. In addition to BTC information, the SCS also retrieves information on which hosts have the split-DSI module deployed and therefore have the capability of acting as a proxy. If it is determined that splitting the connection at one or several points is beneficial, such as at point B, the SCS responds with the URL(s) of the appropriate split-server(s). The client then pushes its file to a split server at B requesting the data be forwarded to the actual destination C. In a similar manner, the client can pull a file from host C by requesting the server B to

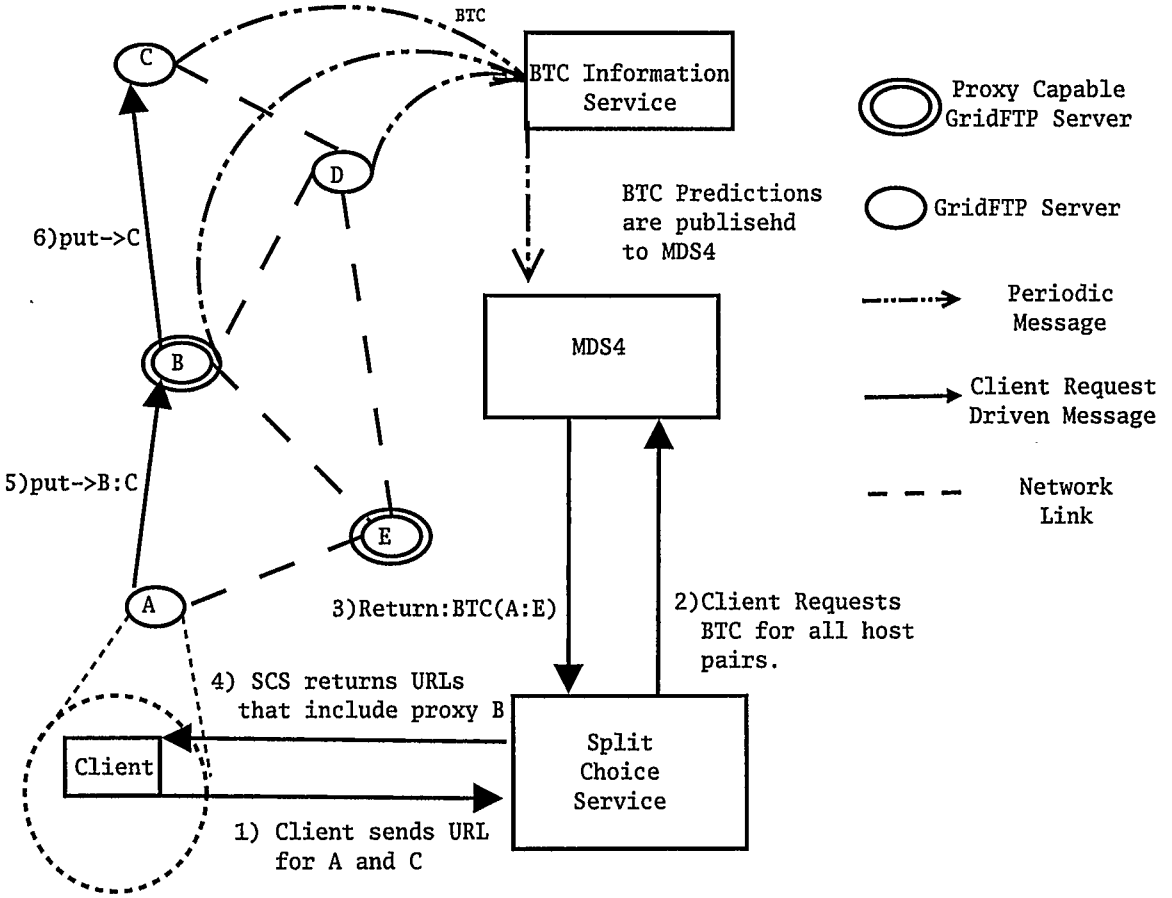

Figure 4.2: GridFTP Overlay Network Architecture

retrieve and forward the file to it. Routing information of the split-TCP connections is included in the URL passed to the split server.

4.3 Split-Data Storage Interface Module

The GridFTP server distributed with the Globus Toolkit v4 provides hooks to write a Data Storage Interface (DSI) [23]. The DSI allows developers to implement modules that know how to read and write from special storage systems. The DSI is passed

GridFTP requests such as *put, get* and *stat* and responds to them in a way specific to the storage device in question. A server can be configured to use only one DSI or the DSI can be dynamically loaded when a client requests it with the appropriate command.

The split-DSI module implements a split-TCP connection on a GridFTP server by forwarding data from *get* and *put* requests to other split servers or a normal GridFTP server. The overlay module handles buffering of the data between connections. This effectively splits the GridFTP connection into two or more TCP connections, buffering data between the two connections. In addition to buffering the data between connections, the split-DSI module handles the delegation of credentials, negotiation connection attributes such as buffer sizes, as well as handling other command that are required for seamless integration such as *stat.* To deploy the module, it needs to be compiled and placed in the library path of the GT4 GridFTP server. Clients will then be able to request the split-DSI functionality. A client not designed to explicitly request DSI functionality may still use a proxy server in which split-DSI functionality is the default.

Conforming to the GridFTP protocol allows the overlay network to leverage the GSI single signon functionality. Each GridFTP server can be delegated a credential to act on behalf of the client. A proxy then uses the credential to connect to other GridFTP servers. This allows fine grained access control to the proxy servers without constant user intervention.

The Split-DSI module gets routing information from the URL it receives from the GridFTP client. For example if a host wishes to retrieve a file located at gsiftp://host1.domain1.ca//file1 by using a GridFTP proxy server at host2.domain2,

it would submit a *get* request for the URL:

gsiftp://host2.domain2/gsiftp://host1.domain1.ca//file1

The prefix gsiftp indicates that an FTP connection utilizing GSI security is requested. Alternatives to this are ftp, which could be anonymous. Whether or not a server accepts an anonymous connection is a policy decision. Using the URL to pass proxy information is very important because it allows seamless integration with current clients. The split-DSI module has been successfully used to split GridFTP connections with *globus-url-copy*, unmodified GridFTP servers (as endpoints) and the Reliable File Transfer Service (RFT) [56]. As described in section 2.1, RFT is a service that allows users to submit a set of file transfers for the service to complete on the users behalf in a fault tolerant manner. *Globus-url-copy* is the GridFTP client distributed with the Globus Toolkit. The integration with current clients is critical for the GridFTP overlay networks usability. Many technologies are not adopted because they would require modification of numerous systems, workflows and systems already in place.

4.4 Bulk Transfer Capacity Information Service

The BTC information service provides a prediction of achievable throughput, as opposed to available bandwidth. Achievable throughput is the throughput that can actually obtained by an application. This is affected by a variety of factors including characteristics of the link such as latency, maximum transmission unit and packet loss. In addition to network characteristics, throughput can also be affected by disk access speeds and the load on the computer.

4.4.1 TCP Throughput prediction

The BTC service provides the prediction of achievable throughput required in order to make decisions on when and where to split TCP connections. There exist two possible approaches to predicting BTC. One can base it on the underlying characteristics of the network and a model, or one can base it on past experience of the applications. Previous work has termed this distinction formula based (FB) versus history based (HB) prediction [42].

Utilizing underlying characteristics of the network links involves getting information such such as MTU, RTT, loss rates, total bandwidth and available bandwidth. From these metrics, an algorithm could utilize the various models developed for TCP performance [61, 67, 18] to arrive at reasonable choices of where to split a TCP connection. One difficulty in utilizing this approach comes from getting measurements. This level of detail about every link in the network is often not available. The TCP performance models may be unusually sensitive to errors in the measurement of these metrics. This results from the fact that one part of all the TCP models developed is the ratio :

$$BW = \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \tag{4.1}$$

where MSS is the maximum segment size, RTT is the round trip time, p is the packet loss probability. This model is discussed in section 2.3.3.

This ratio implies that errors in RTT, MSS, and loss probability will compound in the ratio calculation, rather than be additive. In fact it has been shown that formula based predictions of the throughput of a particular flow are often off by orders of magnitude because the flow being predicted will increase the RTT and

loss experienced by all streams [42]. These formulae are also typically based on the assumption that the congestion window is the limiting factor. However, the most significant gains in splitting a connection can often be when this is not the case [74].

Formula based prediction was successfully used by the RON [6]. In this case, a simplified version of formula 2.1 in section 2.3.3 was used to determine if connections would achieve improved throughput. The throughput "score" of a path was given by:

$$score = \frac{1.5}{RTT * \sqrt{p}} \tag{4.2}$$

In addition to simplifying the formula, the loss rate was given a minimum value of 2% because the formula is more sensitive at lower loss rates. This resulted in successfully finding improved paths for TCP although, despite using a very conservative coarse grained approach, a portion of the connections became worse rather than better.

History Based prediction techniques primarily use previous experience of the application in question. Given the difficulty of getting accurate predictions from external tools, utilizing previous throughput actually experienced by applications is likely the most robust approach. In contrast to formula based techniques, they tend to be readily available assuming the application can be instrumented appropriately. Recent work has also indicated that even relatively simple techniques can be quite accurate [42] in general, as well as, for GridFTP in particular [87]. Previous work has shown that even simple predictors such as window based moving averages and medians is very unlikely to result in errors of greater than 25% for larger transfers [87]. GridFTP throughput predictions can be made even more accurate when

used in conjunction with NWS [86]. NWS has been found to be inaccurate on high bandwidth ($\geq 1Gbs$) links when used alone [86, 87]. This indicates that NWS cannot accurately measure bandwidth but does seem to capture the non-stationarity of the network bandwidth.

The GridFTP overlay service uses a hierarchical BTC information service that reports a prediction of BTC based on various simple prediction techniques, such as taking the median, using only GridFTP transfer logs. This provides unobtrusive passive monitoring of the network. The information is then aggregated and disseminated through MDS4 [79].

The BTC Information Service is similar to the system described in [87] which provides reasonable estimates of throughput with errors not exceeding 25% in most cases. It is not the intention of this thesis to suggest new history based prediction techniques but to test old ones with a new data set and utilize this work to select split points.

4.5 Split Choice Selection Service

The SCS determines which proxy servers to use when transferring a file between two sites. The SCS is currently implemented as a script that makes decisions on where to split a connection for a GridFTP client. The script takes as input a source and a destination URL. It then retrieves throughput statistics and the location of proxy servers from the BTC information service and chooses split points if it is predicted that the use of proxies will improve throughput. The SCS then returns to the GridFTP client a modified source URL that will use the correct splits.

When basing split point selection on achievable bandwidth measurements, the algorithm for selecting split points attempts to find the path from source to destination with split points in between that maximize the minimum BTC along the way. This problem has been referred to as Maximin path optimization [83].

The creators of the Logistical Session Layer (LSL) recognized that the Maximin path problem could be solved by building a minimum spanning tree for each host [83]. The authors use a modified greedy tree building algorithm. The authors also partition many hosts into groups with similar bandwidth. This partitioning greatly simplifies the routing tree built and can minimize extraneous hops. This algorithm has a complexity of O(NlogN) for an implementation that keeps BTCs in sorted order. LSL is discussed further in section 3.5.

SCS currently uses a brute force search which is not overly expensive when there is a small fixed number of split points used for the overlay topology, such as the one described in chapter 5. This would be sufficient for many grid applications. If more scalability was required, the LSL algorithm would be more appropriate. The LSL algorithm could be further scaled by creating a hierarchy of servers, all implementing the LSL algorithm, with some using hosts as nodes, others using domains.

Choosing split points in the manner used by LSL and the SCS assumes that maximizing the minimum link capacity will result in a bandwidth somewhat similar to the minimum link capacity with little overhead. The main assumption used is that the streams do not affect one another much. This assumption is not correct if the streams starve each other of resources, such as buffer space at the routers, resulting in larger RTT, or even simply using the same operating system, interfaces or switches at the same time. Emulation and real world experiments indicate that

the two TCP streams of a split-TCP connection do affect each other. This issue is discussed further in chapter 5.

There are many ways to deliver the decision of where to perform split connections to the client. The decision algorithm can be run on the client, on GridFTP servers, or as a separate service. The ideal case would depend on a user's needs and level of control over the computers being used. An individual user is, in some cases, more likely to know which resources are available to them.

It is simplest for a user to provide information on where to split to a client. In the absence of an information service to provide throughput, a client can take advantage of split servers by using a configuration file provided by the user. The paths in the configuration file would be based on the user's or administrator's unique knowledge of the network. A separate service makes it easier for a virtual organization to aggregate the information, such as BTCs of links they use and split servers the group members have access to. In many cases the users of the system would make similar transfers, meaning the decisions on where to split connections and the results of these decisions could be cached. A separate service will also likely scale better because the SCS must retrieve BTC predictions for all host pairs from MDS4. If the system became large, a separate service would more easily be able to cache this information than clients. The disadvantage of a separate service is that it must be made available. If the network between the client and server was down clients would have to rely on caches of previous queries or simply not use split servers. The decision on where to split connections can also be made at GridFTP servers themselves. A client could simply query a nearby GridFTP server for a file, and that server would go directly to the source or use splits if necessary. The main advantage to having the GridFTP server

make the decisions is that unmodified GridFTP clients could use it by indicating the source server in the URL. The disadvantage of using the GridFTP servers in this way is it necessitates one split even when it would not improve performance. Instead of automatically going to a nearby server, the GridFTP protocol could be extended by the source server suggesting a split point when it receives a *get* or *put* but this would entail modifying the GridFTP client.

4.6 Proxy Placement and Selection

The use of a split-TCP service to improve data transfers requires the solution of two distinct problems: proxy selection and proxy placement. Proxy selection involves the dynamic selection of split points given the source, destination, a set of proxies and current network conditions. In the GridFTP overlay network, this decision is made by the SCS. Proxy placement addresses the problem of where to deploy proxies in the first place. Proxy placement determines where proxies are most likely to be advantageous or useful. This would likely be influenced by traffic patterns and network topology.

The simplest choice for proxy placement is to simply put a proxy server everywhere there is a GridFTP server to host it. Under this deployment technique some GridFTP servers may get more requests to act as a proxy than can be efficiently served by the host. The negative effects of this can be minimized by limiting the number of proxies run at any one time. The fact that a specific proxy server is over-utilized is also a useful signal for both network and system administrators. System administrators may wish to add more resources at that location. Network adminis-

trators may wish to examine the causes of such high utilization of the split server. For example, it may indicate a machine that has poorly configured TCP buffers.

In order to predict where proxies might be useful, models such as equation 2.1 are useful. These models indicate that at any point in which there is likely to be a significant beneficial change in MSS, or loss probability on a long latency link, it may be beneficial to have a proxy server located there.

4.7 Summary

The GridFTP overlay network improves a GridFTP transfer by splitting the transfer into several TCP connections in series. The information services that provide the proxy selection decision utilize only passive observations and are extensible allowing users to develop more accurate predictors if active probing is available. The proxy functionality extends the widely deployed GridFTP server distributed with the globus toolkit. This allows the proxies to be used by unmodified clients and greatly simplifies deployment and allows fine grained access control to the proxies. Without the ability to use unmodified clients, the overlay network would not likely be adopted by those dependent on legacy services, workflows or applications.

Chapter 5

Performance Results

This chapter presents the results of experiments that test the potential effectiveness of the GridFTP overlay network in improving throughput and link utilization. The split-DSI module's performance as proxy server is first evaluated independently in an emulated environment to test its response under various network conditions. The second section presents the performance of a GridFTP overlay network deployed at various sites across North America.

5.1 Emulation Environment

All emulation experiments were run on a cluster of HP Proliant DL145 G2 and DL585 servers connected by switched Gigabit Ethernet. Each DL145 has two 2.2 GHz Opteron Processors and 2 GB of RAM. The DL585 has four 2.2 Ghz processors with 4 GB of RAM. The DL145s were running the Fedora Core 2 version 2.6.10-x_FC2smp kernel. The DL145s were used as clients in the experiments. The DL585 was running SUSE Linux 9.1 with the v2.6.5-7.151-smp kernel.

The DL585 was used to host the Internet Protocol Traffic and Network Emulator IP-TNE [15]. IP-TNE is a parallel discrete-event network simulator enabled to emulate traffic in real time. The emulator interacts with real hosts by acting as a router for the real hosts. When the emulator receives a packet from a real host, the packets traversal through the emulated network topology is simulated. If and when

the packet is intended for another real host the packet is released to the real network at the appropriate time.

Using the emulator enables experiments to be conducted in a variety of network topologies while still using the actual protocol stack and application being evaluated. One of the hazards of using the emulator is that it uses a simplified model for various parts of the network. For example the router model in the emulator will not behave exactly as a router in the real world. However, the model is assumed to capture the most relevant aspects of the system. Even if the model does not match the real world exactly the emulated experiments still provide an understanding of how the system responds to changes in the parameters being modified.

5.2 Emulation Experiments

5.2.1 Goals

The goal of the initial emulation experiments is to test the change in goodput of split-TCP connections under various network conditions. According to formula 2.1, a reduction in RTT by half should increase the throughput by a factor of two if the transfer is still congestion window limited and the loss remains the same. The goodput achieved is still constrained by the bandwidth available, but the utilization of available bandwidth should increase. In addition to this, the response to maximum transmission unit (MTU) is observed.

5.2.2 Metrics

This set of experiments has two metrics; transfer time and goodput. Transfer time is defined as the total time required for a file transfer including connection setup and teardown. The time taken is the time elapsed from the moment the file transfer client is invoked to the moment it exits. Goodput is defined as the file size divided by the transfer time. Overhead is also discussed in these results. Here, overhead is defined as the difference in goodput between a single connection over the bottleneck link and a split connection that includes the bottleneck link. The bottleneck link of a split-connection is the link with the lowest available bandwidth. For example, given two links, AB and BC, in which a connection over AB has a goodput 50 Mbps and a connection over BC achieves a goodput 100 Mbps, the split connection utilizing both links should also have achieved a goodput of 50 Mbps; the same as the connection for AB. If the goodput of the split connection is 40 Mbps, the overhead is 10 Mbps.

5.2.3 Topology

The topology of the emulated network and the hardware setup for this set of experiments is shown in Figure 5.1. Clients A, B and C where equipped with GridFTP [2] servers and gsissh and globus-url-copy packaged with the Globus Toolkit 4.0 [33]. All hosts were configured to route packets from the 10.x.x.x domain to the emulator. Since all experiments had a maximum latency of 100 ms and maximum bandwidth of 100 Mb per second the maximum possible bandwidth delay product was approximately 20 Mb or approximately 2.40 MB. This is an approximation because the round trip time will be slightly higher than twice the latency due to delays at the router. The hosts were configured with maximum write and read buffers of 8MB

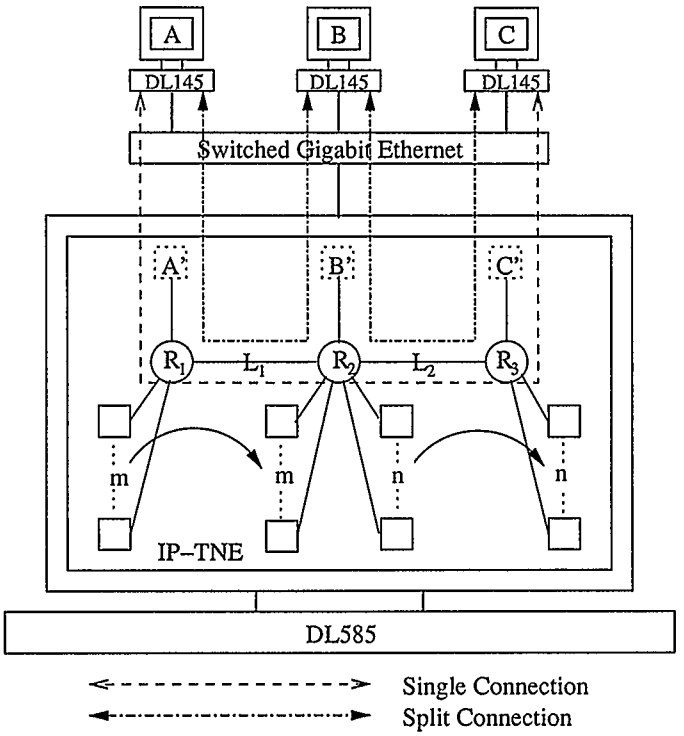

Figure 5.1: Simulation model [74].

and transfers were done using 4MB TCP buffers ensuring the receive window did not limit the throughput of the connection.

Transfers were conducted from host A to host C. A split-DSI module was installed on the GridFTP server at host B and was used as a proxy for the GridFTP transfers. The latency (δ) and MTU of links L_1 and L_2 and the number of simulated traffic streams m and n were changed to explore the parameter space. Over L_1 the GridFTP transfer had to compete with m sources transferring to m destinations. Over L_2 the GridFTP transfer had to compete with n sources transferring data to n destinations. The total capacity of both links was set at 100 Mbps and the MTU size was maintained at 1500 unless otherwise stated.

The competing traffic is modeled as on/off traffic sources. The amount of competing traffic was varied by changing the number m and n source and destination hosts. The duration of on and off periods followed the Pareto distribution. The Pareto shape parameter was set to 1.4 and the Pareto scale parameter is such that the average on or off period was 1.5 seconds. During an on period, packets were sent according to a Poisson process with a mean time between generation of packets such that the average on rate was 5Mbps. This form of traffic was chosen to ensure a known amount of average background competing traffic. This allows the overhead to be calculated.

Table 5.1: Experimental parameters and levels.

Parameter	Levels
$\delta_1 : \delta_2$	10:90, 30:70, 50:50, 70:30, 90:10
L_1 background utilization	10%, 30%, 50%, 70%, 90%
L_2 MTU	1500, 9000

Table 5.1 summarizes the parameters and traffic levels used in the experiments. In all experiments a 4GB file was transferred 5 times. This file size was chosen to provide results that were not affected by slowstart or caching. Three sets of experiments were conducted. The first experiment varied the ratio of latency ($\delta_1 : \delta_2$) while keeping the overall latency between both links at 100 ms. δ_1 and δ_2 are the latency of L_1 and L_2 respectively. In this experiment the background link utilization of L_1 and L_2 by background traffic was kept at 50% and 0% respectively.

The second set of experiments varied the background utilization of L_1 from 10% to 90%. During this set of experiments, the latency of L_1 and L_2 was kept at 10 ms and 90 ms respectively.

In the third set of experiments, the MTU used across $L2$ was varied while using a large number of topologies drawn from the first two experiments.

When performing experiments using an emulator, the emulator may fall behind in the simulation of packets destined for real hosts. In order to ensure this did not materially affect the results, the emulator was instrumented to monitor if it delivered packets on time. Both average and maximum lateness over an experiment were monitored. On average packets were sent early instead of late. For all tests, the maximum average earliness of packets was 7 μs. The maximum lateness of delivery of any packets on one test was 7 ms. The RTT of any TCP connection during the experiment with a maximum lateness of 7 ms was a minimum of 100 ms so the maximum lateness was only 7% of the RTT. The maximum lateness of all other connections was less than 5 ms.

5.2.4 Experimental Results

This subsection compares the performance of GridFTP using a single TCP connection and split TCP connections. Three sets of experiments were run, to examine the effect of latency, the effect of packet loss, and the effect of MTU.

Latency Results

The first set of experiments examines the effects of latency on the performance of the GridFTP file transfers involving split TCP connections. The end-to-end latency was kept constant at 100 ms, but the ratio of the latency of L_1 and L_2 was varied. For example a 10:90 ratio indicates that the latency of L_1 was 10 ms and the latency of L_2 was 90 ms. L_1 was the bottleneck link with link utilization of 50% due to

background traffic. No background traffic was run over L_2. Under the same network conditions, a transfer was made from host A to host C both with and without using a proxy (IE. using a split connection) at host B . Transfers were also made from host A to host B and from host B to host C for comparison.

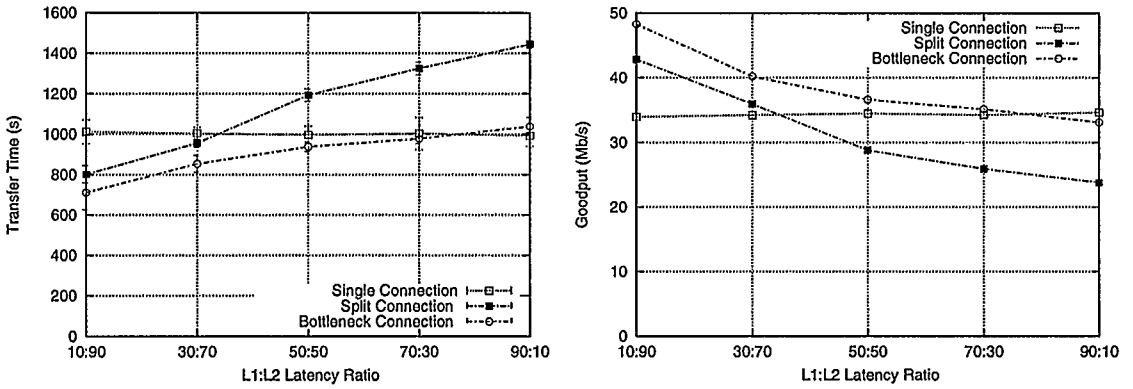

Figure 5.2: Plots of transfer time (left) and goodput (right) versus $L_1 : L_2$ latency ratio.

Figure 5.2 shows plots for the transfer time and the goodput versus the latency ratio of links L_1 and L_2 for 4 GB file transfers. The results indicate what would be expected from the analytical models. The analytical models use RTT instead of latency but RTT is usually well approximated as being twice the latency of the links. Isolating a busy link with a split connection provided a significant improvement over a single connection when the round trip time was kept short on the busy link. When the background link utilization of the first link was 50% and the latency ratio of the first and second connection was 10:90, the utilization of the remaining available bandwidth for the split connection was approximately 86%. This compares with approximately 68% utilization of remaining bandwidth in the case of a direct con-

nection. The longer the busy link the less significant the improvement in utilization of remaining bandwidth. There also appears to be significant overhead in the split connections when the round trip time is not significantly reduced for the congested link.

Link Utilization

The second set of experiments examines the effects of background link utilization on the performance of GridFTP file transfers involving split TCP connections. In this section background utilization refers to the link utilization of background traffic. The background utilization of L_2 is kept constant at 0% but the background utilization of L_1 is varied from 10% to 90%. A link latency ratio of 10:90 is used with latency on L_1 being 10 ms and latency on L_2 being 90 ms.

Figure 5.3 shows plots for the transfer time and the goodput versus the background utilization of L_1. The results of these experiments demonstrate where a

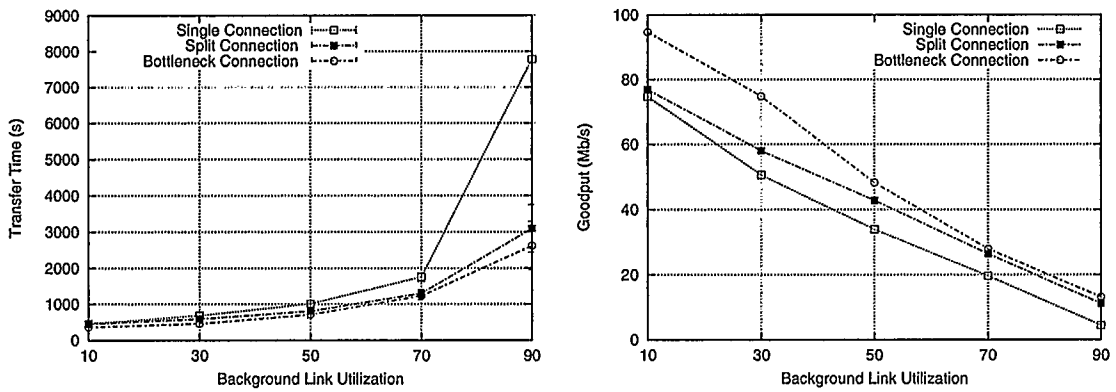

Figure 5.3: Plots of transfer time (left) and goodput (right) versus background utilization of L_1.

split connection can be most beneficial. As the background utilization over the first

link was increased, the goodput gain increased significantly. At 70% background utilization, a split connection achieved a goodput of 26.4 Mbps compared to the performance of a single connection of 19.6 Mbps. This represents a 35% improvement in goodput and link utilization. When the background utilization is 90% the utilization of the goodput of the split and single connections were 11.1 Mbps and 4.42 Mbps respectively; an improvement of 151%. It is also clear from both the latency and background utilization experiments that the performance of the split-TCP connections track the performance of the bottleneck link closely although there is significant overhead. This indicates that the bottleneck link can be used to predict the performance of a split connection, but overhead must be taken into account.

MTU Results

The third set of experiments examines the effects of MTU on the performance of GridFTP file transfers involving split TCP connections. For these experiments, the MTU of the second connection was changed to use jumbo frames of 9000 bytes and the experiments were repeated. The average change in throughput on split connections when compared to split connections in which the bottleneck link did not use jumbo frames, was approximately 3% in most cases. There was a similiarily small increase in throughput on single connections over the bottleneck link. An examination of traces of the links revealed that connections with jumbo frames incurred 5-6 times more packet loss. Increasing the MTU of a connection means TCP will increase its instantaneous throughput more aggresively during congestion avoidance. This can have the effect of increasing throughput but can also increase loss. The increased packet loss cancelled most of the advantages of using jumbo frames.

Jumbo frames typically improve throughput on high-bandwidth, high-reliability networks [64]. They have a less predictable effect on slower networks.

Variance Results

Table 5.2: Standard Deviations for experiments.

RTT		Link Utilization		Standard Deviation % of mean		Standard Deviation	
L1	L2	L1	L2	Single	Split	Single	Split
10	90	50	0	4.69	4.28	47.53	34.33
30	70	50	0	2.34	1.75	23.50	16.75
50	50	50	0	3.52	2.02	35.06	24.12
70	30	50	0	6.34	1.86	63.58	24:71
10	90	10	0	15.34	1.49	70.57	6.66
10	90	30	0	2.08	1.98	14.10	11.74
10	90	70	0	5.53	6.80	96.84	88.39
10	90	90	0	18.39	17.01	1430.05	526.16
90	10	0	30	2.25	1.61	15.35	9.10
90	10	0	50	4.90	3.56	49.32	25.54
90	10	0	70	5.96	7.73	105.89	86.55
90	10	0	90	19.38	16.72	1503.42	472.39
10	90	0	50	4.74	2.72	46.92	25.31
30	70	0	50	3.20	1.96	31.70	16.79
50	50	0	50	3.83	2.39	38.41	19.86
70	30	0	50	1.82	1.51	18.21	11.91

Table 5.2 summarizes the standard deviations of the transfer times observed in the split and single connection transfers done for the experiments. All trials for a given set of parameters were run the same number of times for split and single connections. For most of the experiments, the standard deviation in absolute terms as well as a percentage of the mean is smaller for split connections. This is the case for most experiments including ones in which there was no improvement in throughput. This implies split connections often have the advantage of being more predictable in

addition to having improved throughput and network utilization. A more predictable transfer time could be beneficial to meta-schedulers in planning data movement and data processing tasks. Whether the magnitude of the improvement is enough to be useful would require significantly more study. The utility of the lower variance would be based on the size of the transfers and the behavior of the meta-scheduler.

5.2.5 Overhead experiments

The results from the experiments examining the effects of latency and link utilization indicated a significant amount of overhead in a split connection. There are a wide variety of reasons for overhead to be present. As discussed in section 3.5 there may be a dependency between the first and second links. This is more likely to occur on larger chains of split connections. This section presents an analysis of the overhead observed in the emulated environment. This analysis presents a series of plots of the TCP connections made by using data taken by running *tcpdump* while performing transfers over the emulator using the topology shown in Figure 5.1. The latency of L_1 and L_2 were both set to 50 ms. The utilization on L_1 was 50% with no traffic on L_2. Several other configurations were examined and the traffic patterns were found to be similiar.

Figure 5.4 shows a plot of the data connection of a single transfer from host A to host B as seen from host A. The plot shows the data packets sent and cumulative acknowledgments received over time. The y-axis represents the sequence numbers of the data being sent or acknowledged and the x-axis is time. The result is a smoothly advancing window, with packets sent as acknowledgments are received. In comparison, figure 5.5 shows the data connection of the connection from host A to

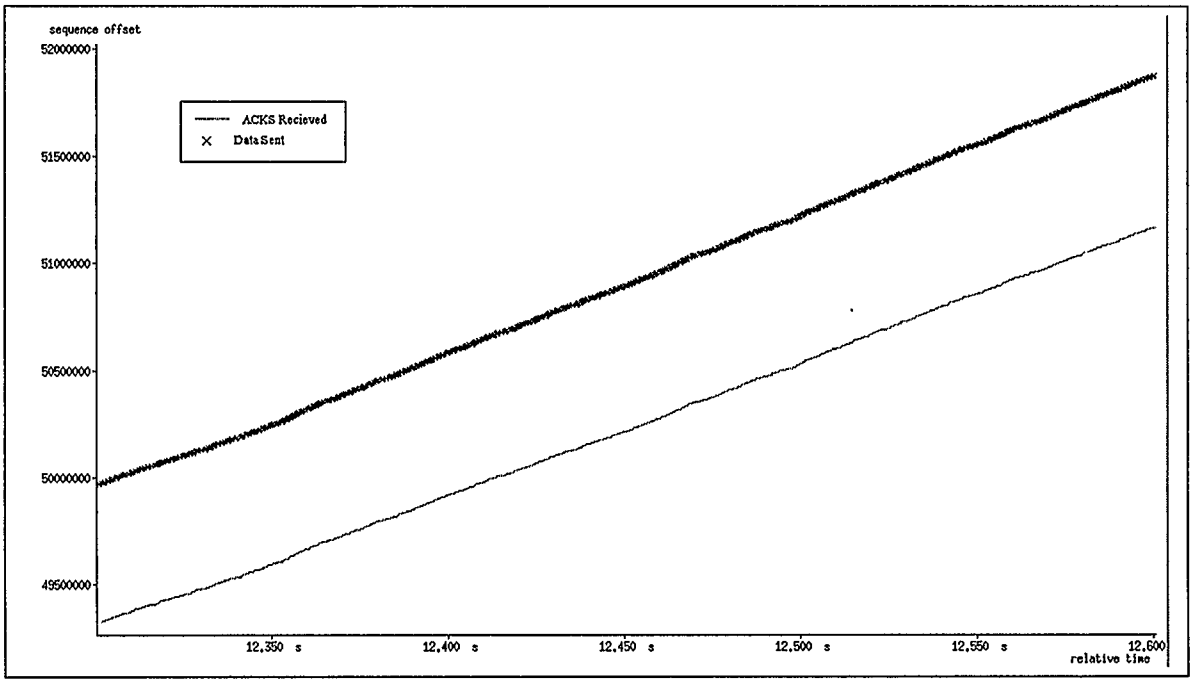

Figure 5.4: Trace of single connection from A to B

host B of a split connection from host A to host C using host B as a proxy. The result is a clearly bursty connection. The acknowledgments are received in rapid bursts. These bursts appear as vertical lines in the acknowledgments received line. In response, the connection rapidly releases new data packets to keep the congestion window full.

The behavior of the link for the split connection resembles ACK-compression [93, 88]. ACK-compression describes a state in which two or more TCP connections transferring data in opposite directions share a queue or buffer. The acknowledgments from one connection become queued behind the data packets of the other connections. When the packets are finally released by the queue, they are released at a faster packet rate because acknowledgments tend to be smaller than data pack-

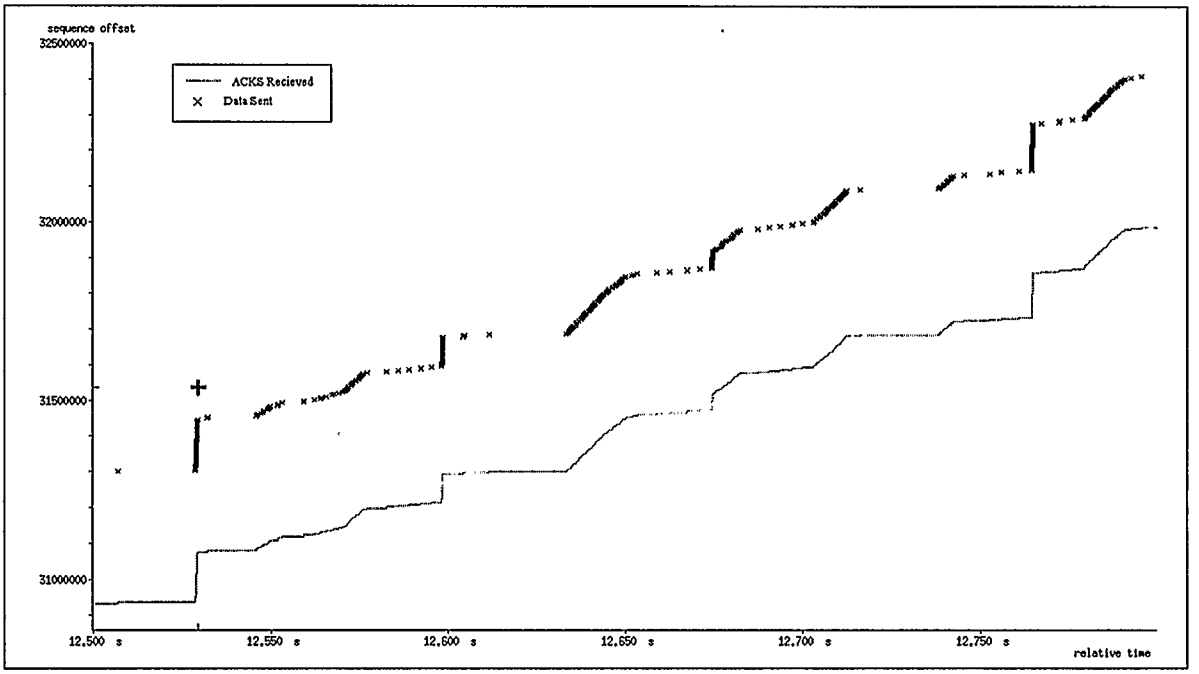

Figure 5.5: Trace of A to B link of split connection

ets. This change in the packet rate means the ACK packets no longer serve as a reliable clock in steady state [93]. The end result is bursty traffic and idle periods resulting in reduced throughput.

ACK-compression has long been observed on two way traffic. In [93] it was observed that ACK-compression can occur on a simple topology in which there are two TCP streams transferring data in opposite directions across the whole connection. It has also been observed when the two streams share buffers at the end points [54], the case here. ACK compression was found to exist in simulations of larger topologies [93] and in traces of live networks [88]. ACK compression is reduced with a delayed ACK strategy (acknowledging every nth packet), but this only has a significant effect with smaller window sizes [93]. In high performance computing environments, it is of-

ten the case that there is very little competing traffic. In this environment, a split connection can cause two large TCP streams with very large window sizes that can experience ACK-compression.

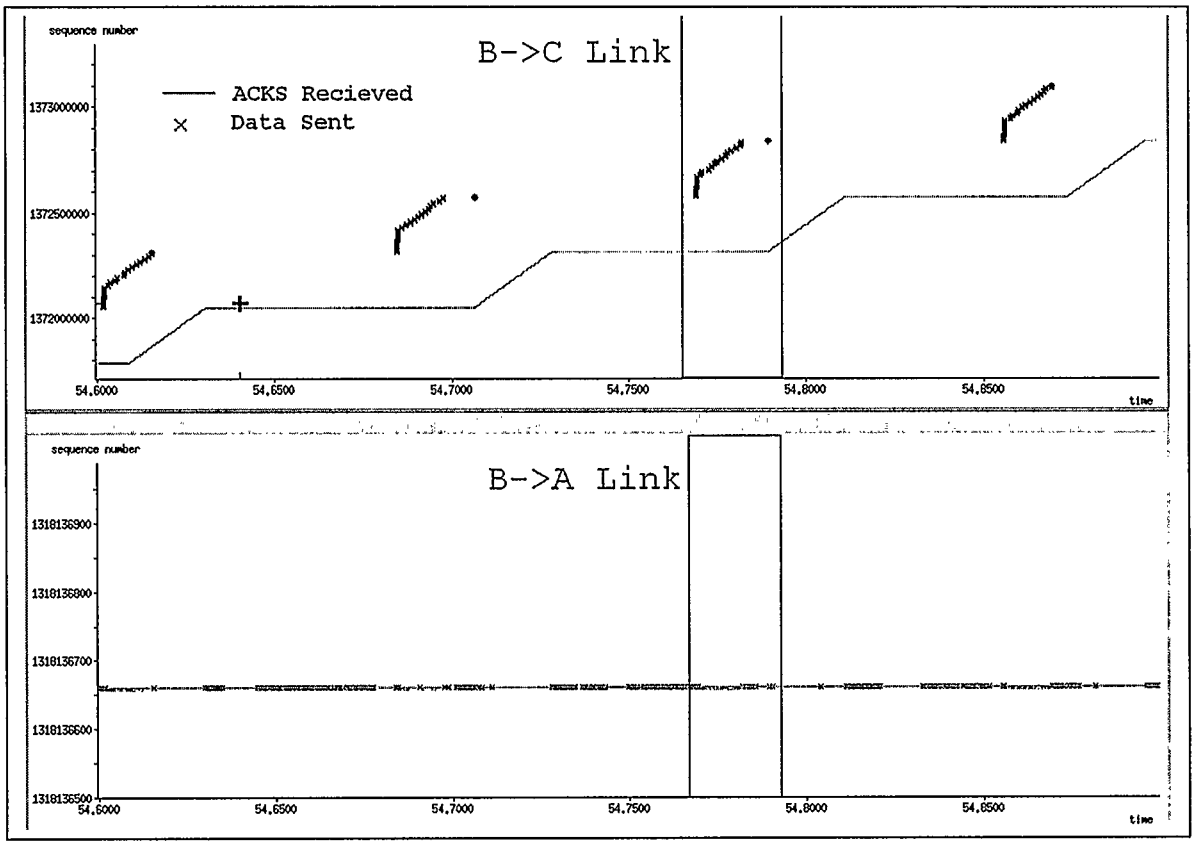

Figure 5.6: Trace of A to B link and B to C link on host B

The main conditions for ACK-compression to occur are that ACK packets are significantly smaller than data packets, and packets from each connection are clustered together [93]. The first condition is true because the connections for these experiments are bulk data transfers. Figure 5.6 shows a plot of both data connections' outgoing traffic for a data transfer using split-TCP as seen from host B. The x

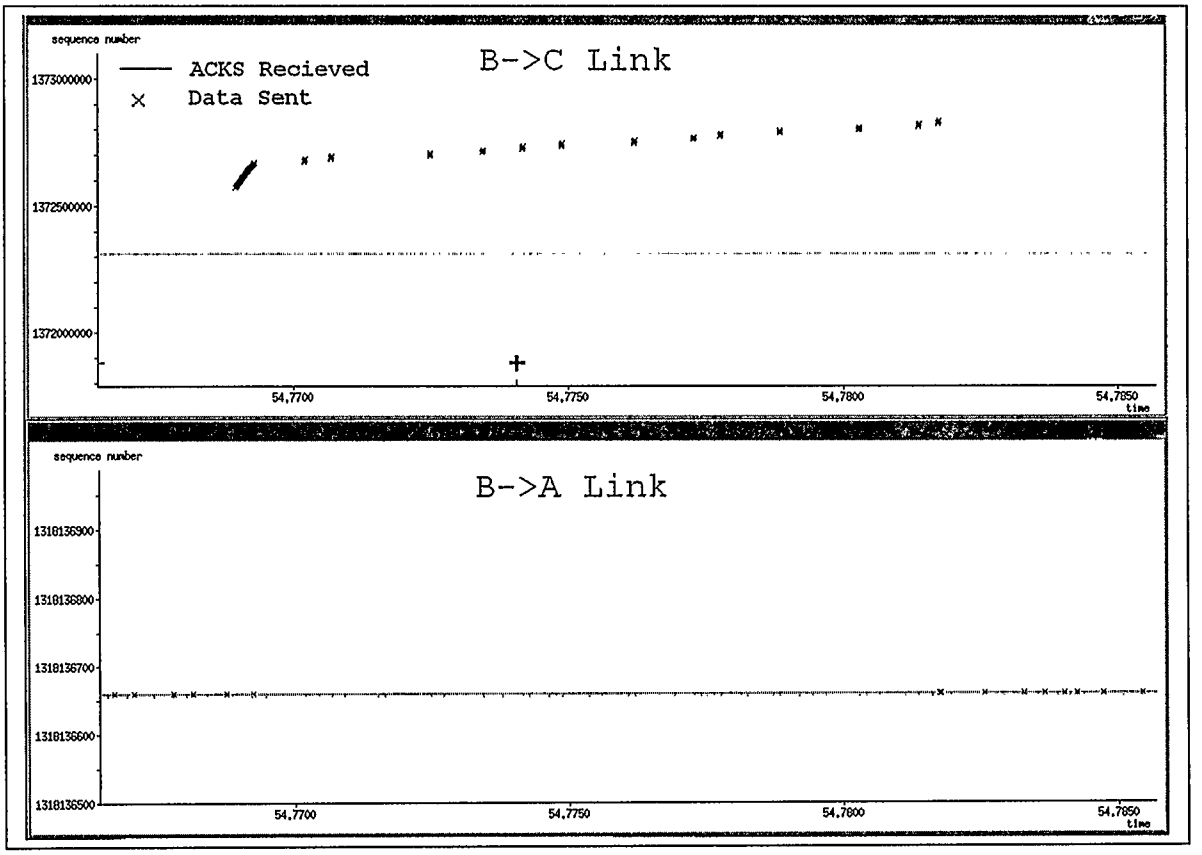

Figure 5.7: Magnification of trace in Figure 5.6

axis (time) for the two plots are aligned. Figure 5.7 is a magnification of the shaded region in figure 5.6. The data and ACK packets from each connection are clearly clustered rather than interleaved. The increased burstiness that results from the ACK compression increased the average RTT of the AB connection from 106.3 ms to 111.3 ms and the maximum RTT from 114.3 to 161.8 ms.

In order to further ensure that ACK-compression was a result of contention for the link between host B and the router, the link speed from host B to the router was changed from 100 Mbps to 200 Mbps. Figure 5.8 shows the effect on transfer

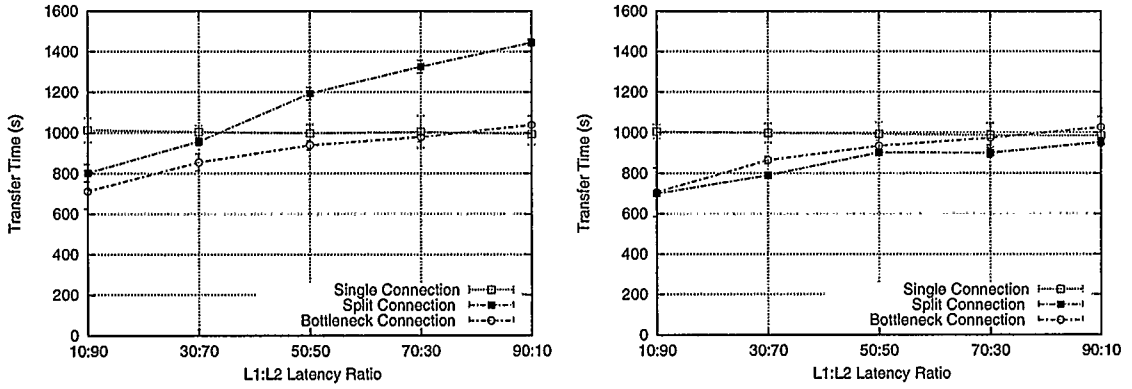

Figure 5.8: Plots of transfer time with link from host B to router at 100 Mbps(left) and 200Mbps (right) versus $L_1 : L_2$ latency ratio.

time of changing the speed of the link connecting host B to the virtual router while changing the latency ratio of L_1 and L_2. The overhead is completely eliminated at all configurations. The new A trace of the first connection is shown in figure 5.9. The burstiness is eliminated and the plot of the trace is similar the trace of a single connection. Goodput and RTT statistics are similar to the those of the single connection as expected.

The IP-TNE emulator holds on to packets from real hosts and paces their release according to the emulated link speed that the real host was given. In order to ensure this was not materially impacting results, the packets were paced at the egress of the kernel, using the traffic shaping software *tc* [44] that is included in most distribution of Linux. There was no change in the behavior. Figure 5.10 shows a trace of the first connection as seen from host B. Because this trace was taken on host B, the data should appear to be acknowledged immediately under normal conditions, but there are clear cases in which the acknowledgments are held back. These appear as

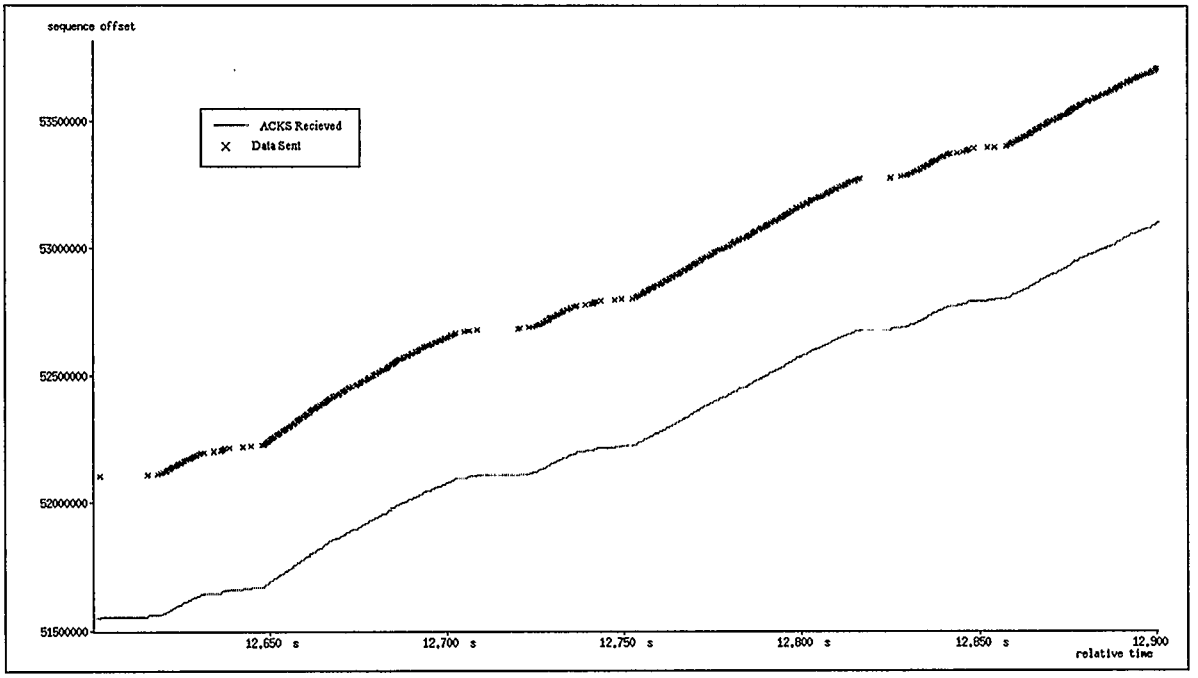

Figure 5.9: Trace of A to B link of split connection with increased forward bandwidth on host B

the places where the data and the acknowledgment lines form right angle triangles. These triangles can be seen at times labeled 04.85 and approximate 04.94.

A split connection will always create two way traffic, but the existence of the two way traffic does not always cause overhead. Figure 5.11 shows the goodput in a similar topology as the previous experiments, changing link utilization and latency with the second connection as the bottleneck instead of the first. The overhead in a split connection disappears when the second leg of the connection is the bottleneck. Further examination revealed that the first connection still suffers ACK compression, while the second one does not. However, since the first connection is no longer the bottleneck, overall throughput is not affected. This indicates the send queue on the

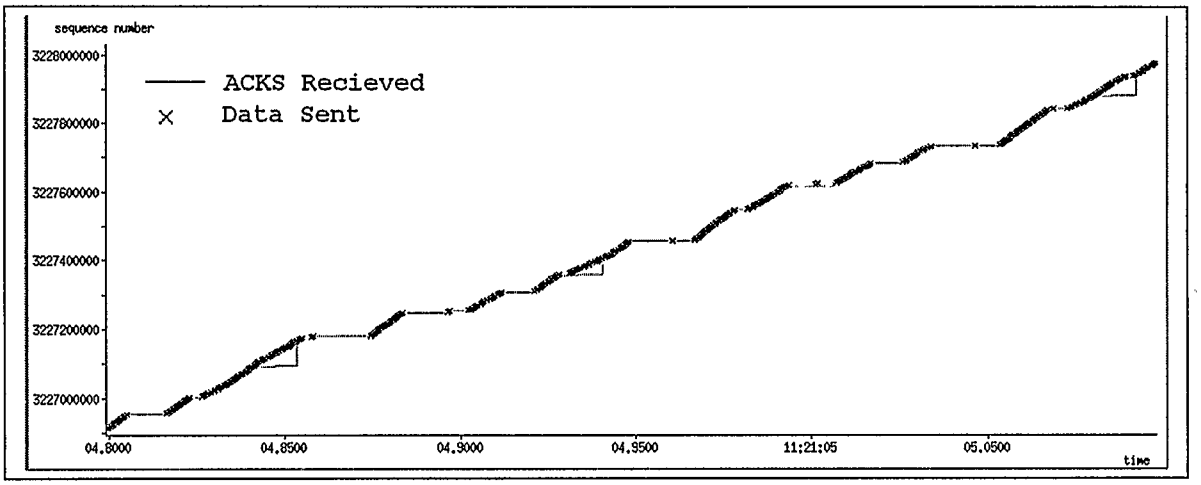

Figure 5.10: Trace of A to B link of split as seen from B host

proxy host is causing the ACK-compression.

ACK-Compression does not seem to have been commented on in the literature regarding split-TCP, possibly because the shared buffers will only occur when a host, rather than a router is used for the proxy. Much of the literature suggest that the proxy behaves as a router to the extent that the traffic does not have to travel over the same link twice. When the proxy acts as a router, there will be no shared output queues.

The fact that the ACK-compression problems only occur on the outgoing connection on the proxy indicate it may be possible to avoid it by modifying the way packets are handled at the host. Prioritizing the acknowledgment in the egress queue of the proxy host will likely alleviate the situation, for example. However changes in the way packets are handled can have unintended side-effects. For example, it was pointed out in [51] that prioritizing acknowledgments can starve the low bandwidth link in the case of two way TCP connections with asymmetric links.

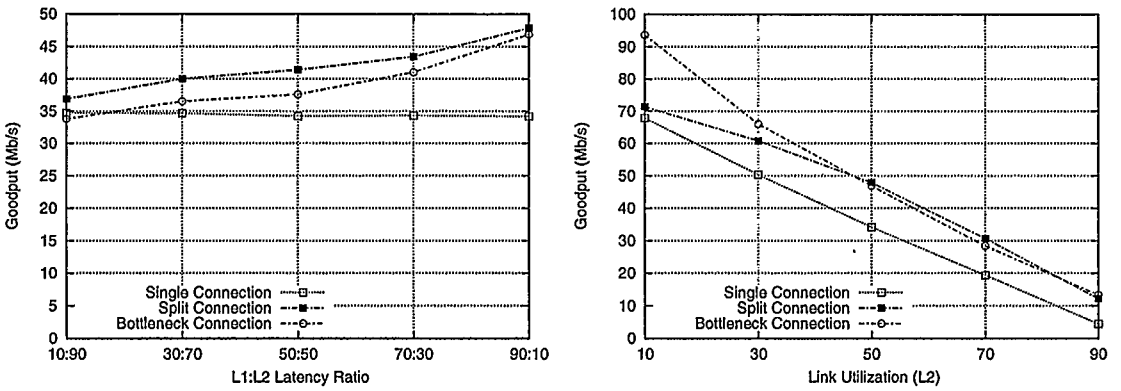

Figure 5.11: Plots of goodput with 50% link utilization on L_2 while varying $L_1 : L_2$ latency ratio (left) and $L_1 : L_2$ ratio of 90:10 with varying link utilization of L_2

The results in which the second connection is the bottleneck also show that the split connection is faster than a single connection that only goes over the bottleneck link. The advantage was statistically significant but less than 10% of the overall throughput. Tests indicated that this was the result of the fact that the transfer from host B to host C is memory to disk in the case of the split connection but disk to disk in the case of the single connection.The single connection transfers were just as fast when done from memory (by copying from */dev/zero*).

5.3 Wide-Area-Network Experiments

A GridFTP Overlay Network was deployed at various sites across North America to evaluate the potential performance improvement to be gained by utilization the overlay network in a realistic setting. The GridFTP overlay network makes use of previous transfers to predict the bulk transport capacity (BTC) it can expect over a specific link. The predictions of BTC for a link will not always be accurate. If the historical data used is very intermittent, the accuracy can be expected to suffer even more. In addition to this, the emulated results from section 5.2.4 indicated significant overhead may occur on a split connection. This overhead may make the assumption that the BTC of a split-TCP connection will reflect the minimum BTC of the individual links used invalid. Given these uncertainties, it is important to evaluate the performance of the split-TCP connections chosen by the selection algorithm using network information available. This section describes the overlay network topology and the methodology used to evaluate the performance of the overlay network when using passive monitoring of intermittent data transfers.

5.3.1 GridFTP Overlay Network Topology

The GridFTP overlay network topology used for experiments is shown in Figure 5.12. It comprised ten hosts including three hosts from the University of Calgary Grid Research Centre (GRC), five hosts from Westgrid, one host from the University of New Brunswick affiliated with ACEnet, and one host from the University of Houston, which is part of the HP CCN Grid. WestGrid and ACEnet are high performance computing consortia in Western Canada and Atlantic Canada respectively. The HP

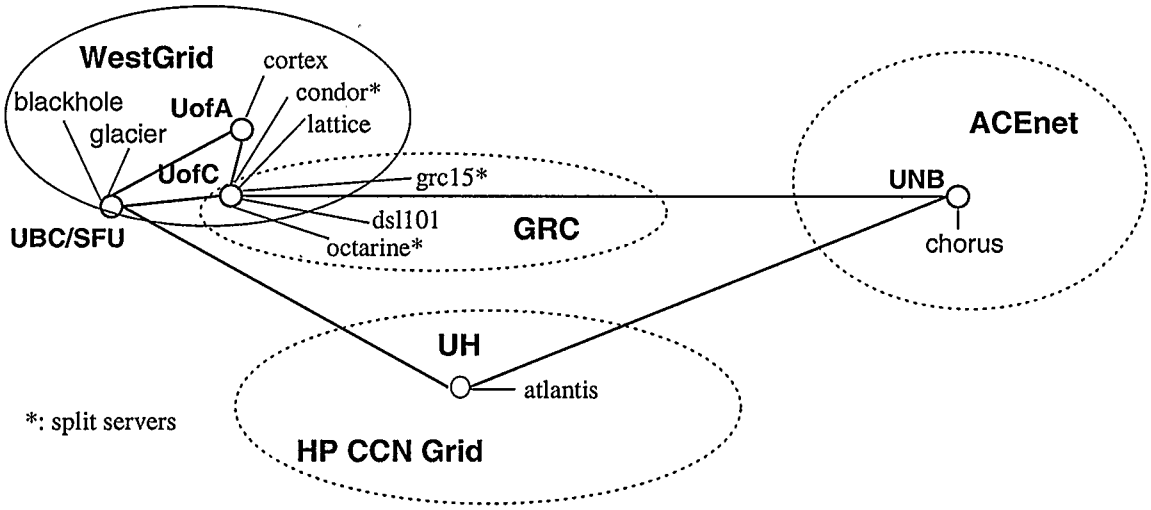

Figure 5.12: GridFTP Overlay Network Topology [75]

CCN Grid is an Hewlett Packard led grid computing collaboration that the GRC is a participant of.

Split servers were placed at hosts grc15 and octarine in the GRC domain as well as condor in the WestGrid domain. Connectivity between domains is provided by the CA*net4 network in Canada and the Abilene network in the United states. The majority of hosts on the overlay network are high performance computing facilities.

5.3.2 GridFTP Log Generation

Initially some of the sites used in these experiments had no GridFTP traffic between them. For this reason, GridFTP logs were created by transferring data at random intervals between the sites. This also allowed us to examine the difference between memory to memory transfers and disk to disk transfers.

The arrival times of the GridFTP transfers were executed as a Poisson process for each source with an average inter-arrival time of 30 minutes and a randomly

selected destination host. These transfers alternated between disk to disk (D2D) transfers and memory to memory (M2M) transfers. Each host had nine alternate hosts to select from with either D2D or M2M transfer performed on average once every hour. This meant that the throughput for comparable transfers between each host pair was measured on average once every 9 hours. For D2D transfers a file size was chosen such that the transfer would take approximately 90 seconds. The range of file sizes available were all powers of 2 from 32 MB to 1024 MB. M2M transfers were accomplished by transferring data from */dev/zero* on the sender to */dev/null* for a period of 90 seconds and recording the amount of data acknowledged by the receiving GridFTP server during that time.

The 90 second average transfer time was obtained by performing transfers between all host pairs and establishing that the throughput achieved was not a function of the transfer time. This was done to avoid the effects of slowstart and the connection negotiation. Transfers with file sizes ranging from 32 MB to 2048 MB were successively transferred between all host pairs 5 times until the transfer took in excess of 240 seconds. The transfers for each host pair was plotted and visually checked for any obvious throughput improvements for longer transfer times. A trend of improved throughput for longer transfers was observed between some hosts. This improvement disappeared at approximately 80 seconds in the worst case. Most of the time the improvement disappeared significantly earlier than that. The time chosen was also limited by the fact that the hosts in the overlay network were production systems. It was important not to utilize too much of the network capacity on these systems.

D2D transfers were intended to include the cost of disk access because not all data transfers are limited by network congestion. In many systems, particularly

those that use high bandwidth networks, the bottleneck for the transfer may occur during reading or writing of data to and from disk. The experiments in this work where conducted on live systems with a diverse group of disk and file systems. Some of the data was transfered from NFS or CFS mounted volumes. The use of caching on NFS and CFS volumes, as well as disk caching done by the operating system, means that what was intended to be a D2D transfer may have in fact been M2D, D2M or M2M.

In order to examine the issues of disk caching, all host pairs were made to repeatedly transfer the same file. This was tried with file sizes ranging from 32 MB to 2048 MB. Successively larger transfers were used until transfers took in excess of 240 seconds. For each file, the median throughput of the subsequent file transfers was compared with the first file transfer in the trial. This gave a measure of improvement on subsequent file transfers of the same file. The median of these improvements for all file sizes was then taken.

Of all hosts, only one had significant, and consistent improvement in throughput on subsequent transfers of the same file. This host consistently had throughput improvements ranging from 75% to 1100% on subsequent file transfers when transfers were to seven other hosts. No consistent improvement was observed on any other host pairs. For this reason, several copies of the files to be transferred were used on the host that demonstrated significant disk caching effects. It was confirmed that this eliminated the caching effects on that host. Due to storage space limitations, a sufficiently large number of files on all hosts that ensured no caching issues were present on any transfer was not possible.

All transfers were performed using a GridFTP client built for the purposes of

these experiments using the API distributed with the Globus Toolkit. This specialized client reported performance information every time it was received from the destination server. Time measurements for throughput were taken by the client using the performance plugin from the GridFTP libraries. Connection negotiation time was taken into account by using the time before connections are initiated as the start time.

5.3.3 Analysis of Split Point Selection

The behaviours of three predictors of bulk transfer capacity (BTC) were collected to test the hypothesis that choice of predictor is irrelevant because the predictors would result in the same choice of proxies most of the time. Data for predictions made using D2D and M2M transfers using a modified exponentially weighted moving average (EWMA), median and moving average.

The median of a data set W with n elements is formally defined as the value $X_i \in W$ such that when all $X_j \in W$ are sorted, an equal number of elements are before and after X_i. If n is even the median value is the average of the two elements $X_i, X_k \in W$ such that an equal number of elements are before X_i and after X_k. Medians were chosen because they are unaffected by outliers.

The modified EWMA was used to respond to network outages more quickly. The usefulness of this quick response is questionable due to the intermittent nature of data points. The modified EWMA is formally defined as :

$$\hat{X}_i = \begin{cases} X_{i-1}, & \text{if } X_{i-1} = 0 \text{ or } X_{i-2} = 0 \\ \alpha X_{i-1} + (1-\alpha)\hat{X}_{i-1}, & \text{otherwise} \end{cases}$$

where α is a tunable parameter balancing noise filtering and response to material changes in network conditions, X_{i-1} is the most recent value collected, and $\hat{X}_{i-1}$ is the previous value calculated. For the purposes of these experiments, $\alpha = 0.3$. The first case in the function definition has the effect of reporting an outage and a return of a server immediately.

The moving average $\hat{A}_i$ is formally defined as :

$$\hat{A}_i = \frac{1}{n} \sum_{k=i-n}^{i-1} X_k$$

where $(X_{i-n}..X_{i-1})$ are the previous n values collected and n is a parameter representing the number of previous measurements that are used in the calculation. The tuning of n balances the response to material changes in network conditions with smoothing out noise.

Moving average and median can both be done with a constant n which changes the amount of time covered by the measurement window, or the span of time in the window can be held constant and n can be changed. Monitoring data was collected based on both methods, however the performance of split connection choices was only tested for metrics based a constant value of n.

At exponentially distributed intervals, the SCS service (see section 4.5) was queried on each host pair to determine if a split-connection was judged to be beneficial. The use of an exponential inter-arrival time means that the measurements form a Poisson process. Under a Poisson process, asymptotically, the proportion of measurements that have a given state is equal to the amount of time the environment spends in that state. A split-connection was judged to be beneficial if the predicted

minimum BTC of the split connection links was greater the predicted BTC of the direct connection. The possible splits were evaluated based on the D2D and M2M throughput predictions using the median of the previous 5 measurements. D2D predictions using a modified EWMA were also tested. On any connection in which a split connection had a higher estimated bandwidth than the direct connection, a split-connection and a direct connection were performed one after the other.

5.3.4 Results

This section presents performance results of the GridFTP overlay network obtained from data collected over a period of three weeks in April 2006. Accuracy of the BTC predictors is presented, followed by performance of the split-TCP GridFTP connections chosen by the overlay network.

5.3.5 Accuracy of BTC Predictors

To compare the error rates of the two linear BTC predictors on a single transfer, percent error is defined as:

$$\% \text{ error} = \frac{|\text{Predicted Throughput - Actual Throughput}|}{\text{Actual Throughput}} \times 100$$

Table 5.3 shows the average percent error of the predicted throughput for all of the actual disk to disk (D2D) direct transfers for the three predictors being examined. D2D transfer estimates for this network have similar levels of accuracy as those found in [87]. This level of accuracy occurs despite using historical data with transfers several hours apart. The large prediction error of the memory to memory (M2M) metric indicates that disk I/O can have a large effect on GridFTP performance and attempts to predict throughput should take this into account.

Table 5.3: Table of Accuracy for BTC predictors

Metric	Average % error	Standard Deviation
D2D Median	22.01%	57.70%
EWMA	22.30%	42.38%
M2M Median	196.31%	738.13%

The large standard deviation is a result of a long tail in the distribution. Figure 5.13 shows a histogram of the percentage error for the D2D Median. 50 % of the measurements have less than 10 % percentage error. The values range up to a of 1600 %. Approximately 1 % of the values are above the range of the graph. Figure 5.13 is representative of the other two predictors.

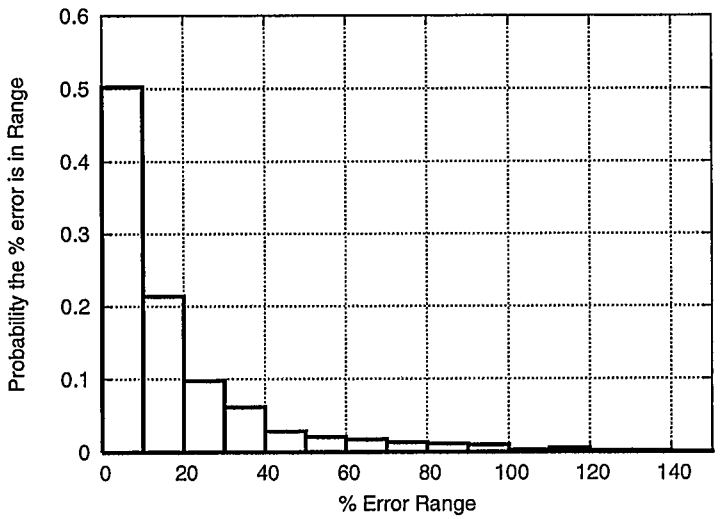

Figure 5.13: PDF of % error for D2D Median

5.3.6 Agreement between Metrics

Table 5.4 shows the level of agreement on where to split connections between the decisions made based on the three metrics. Complete agreement implies the split

point selection algorithm comes to the same decision for which split point(s) to use. Agreement on one split or two splits implies that the three metrics result in agreement on what the optimal split point may be for a particular number of splits, but their predictions on whether or not it would be beneficial may differ. The level of agreement is small even though the performance results of the metrics are quite similar. Complete agreement when the predicted improvement was in excess of 25% was greater, but there still were variations.

Agreement type	% Aggreement
Agreement on 1 split	52%
Agreement on 2 splits	27%
Complete agreement	33%
Complete agreement +25%	52%

Table 5.4: Agreement between Metric choices

5.3.7 Overall Performance of Split-TCP Connections

For the purposes of this work, the predicted or actual improvement of a split-TCP GridFTP transfer was defined as:

$$\% \text{ improvement} = \frac{\text{S - N}}{\text{N}} \times 100$$

where S is the throughput of a split-TCP GridFTP transfer and N is the throughput of a normal single TCP connection GridFTP transfer.

Overall performance of splitting a connection using the three predictors to decide where to split connections appears in Table 5.5. This is the performance of the split connections whenever the SCS predicts even a minimal performance improvement

Table 5.5: Performance of Split-TCP Connections

D2D EWMA	single	double
Number of Transfers	1486	82
Average Predicted % Imp.	66.01	85.13
Average Actual % Imp.	46.27	12.75
95% Conf. Radius	7.13	21.49
Median Predicted % Imp.	27.66	46.08
Median Actual % Imp.	5.27	-4.74
D2D Median	single	double
Number of Transfers	1832	85
Average Predicted % Imp.	64.51	124.11
Average Actual % Imp.	48.15	13.89
95% Conf. Radius	6.16	21.77
Median Predicted % Imp.	21.33	32.00
Median Actual % Imp.	6.22	-12.26
M2M Median	single	double
Number of Transfers	1819	27
Average Predicted % Imp.	136.58	153.75
Average Actual % Imp.	53.45	129.94
95% Conf. Radius	6.66	82.58
Median Predicted % Imp.	42.20	131.78
Median Actual % Imp.	11.80	120.78

and makes no allowance for overhead. The performance of each predictor is divided up between the times it predicted a single split point and a double split points. Up to 14 data-points were removed from the data sets because the predicted improvement of these points was greater than 6.5 standard deviations from the mean; this removed approximately 0.8% of the data points that distorted the predicted improvement average significantly. The value of 6.5 standard deviations was chosen becuase it removed the small group of measurments that clearly represented outliers. The number of datapoints removed varied slightly between predictors because some of

the predictors did not predict an improvement at these moments; they predicted zero throughput instead. These points are included in the rest of the results. The average actual improvement was approximately 48% for the D2D median predictor. Split-TCP connections were used whenever predicted improvement was greater than zero.

The use of two proxies was very rarely predicted to improve throughput and for the D2D transfers, no statistically significant improvement was detectable. This is likely a result of the topology of the test bed. The network latency between the split servers was low when compared to the latency between the edges of the network. In this environment it is unlikely that two split points would be beneficial. The actual improvement of split connections using the M2M predictor was significantly better than the D2D predictors. This is likely because the M2M predictor is not affected by disk effects and can more accurately predict the effects of using two proxies because a proxy to proxy transfer is memory to memory.

The M2M predictors performed slightly better than the D2D predictors for single split points though the improvement is not statistically significant in the case of a single split point. This is likely a result of the fact that M2M predictions only measure the network performance. A predictor that only captures network effects may be more accurate because a proxy can only help overcome failures in the network and cannot change disk effects. This is an indication that integrated network information services such as NWS when available could be useful. The predictive ability of the D2D transfers are more relevant because the data can be collected passively. The two D2D measurements had similar performance. The EWMA predictor had fewer instances in which it was predicted to improve throughput. This is likely because

it predicted BTCs of zero between sites more often, and this would result in no prediction of improved throughput. D2D predictors can be considered more relevant because D2D measurements are readily available from GridFTP transfer logs. M2M measurements would typically require active probing. Because of the similarity in performance of all the predictors, and the greater relevance of D2D predictors, a more detailed analysis of the only D2D median predictor is presented in the remainder of this chapter.

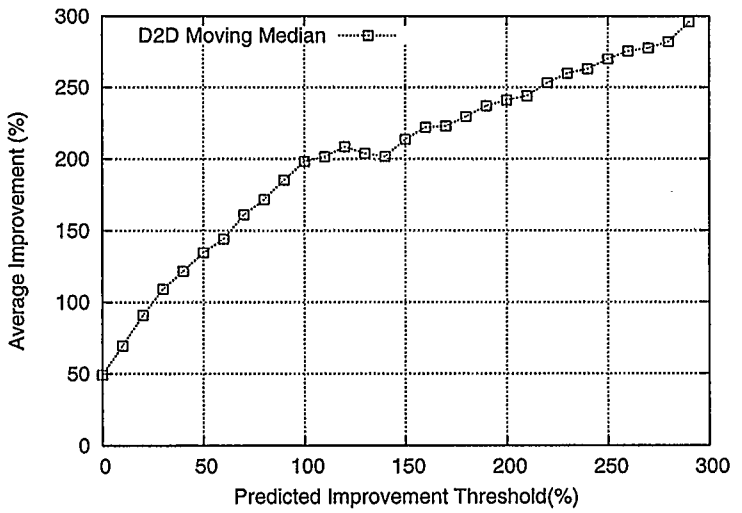

Figure 5.14: Average improvement versus predicted improvement threshold.

Typically, a split-TCP connection would only be chosen if the predicted improvement is greater than a certain threshold. This is to account for overhead. Figure 5.14 shows what the average improvement over direct connections is for split-TCP transfers for different predicted improvement thresholds. The larger the threshold used, the greater the average performance improvement is. For example, if a minimum predicted improvement of 50% is used, the average improvement achieved exceeded 130%.

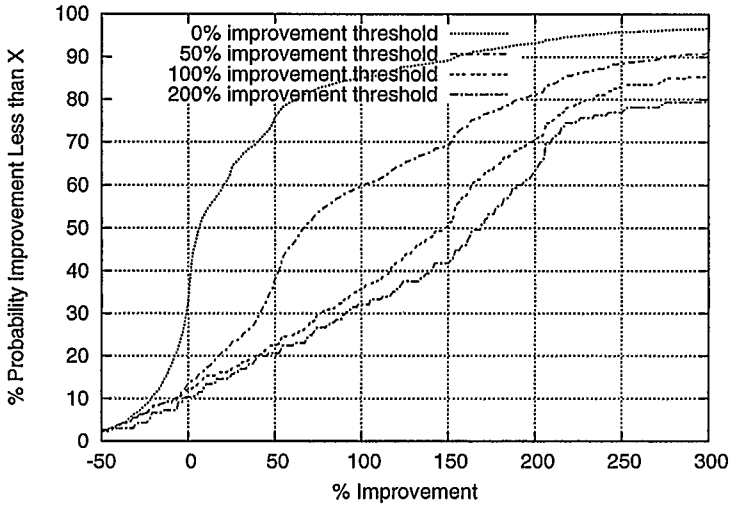

Figure 5.15: CDF of split-TCP connection improvements for various prediction thresholds

Figure 5.15 shows the cumulative distribution function (CDF) of the probability that a split-connection undertaken will have a percent improvement less than some value, given a predicted improvement threshold. The results show that if the prediction threshold is at least 50%, only 13% of the split-TCP connections perform worse than direct connections. In comparison, 33% of the split-TCP connections perform worse if the threshold is 0%. As the threshold is increased beyond 50% the percentage of split-TCP connections that perform worse remains about the same. As such a threshold value of 50% would be reasonable to use.

Figure 5.16 shows a histogram of the number of connections that experienced various ranges of improvement when the minimum predicted improvement threshold was 50%. A majority of split connections had a 30% to 200% improvement in throughput. Only 13% of the connections experienced reduced throughput.

Figure 5.17 shows a histogram of the number of host-pairs that experienced cer-

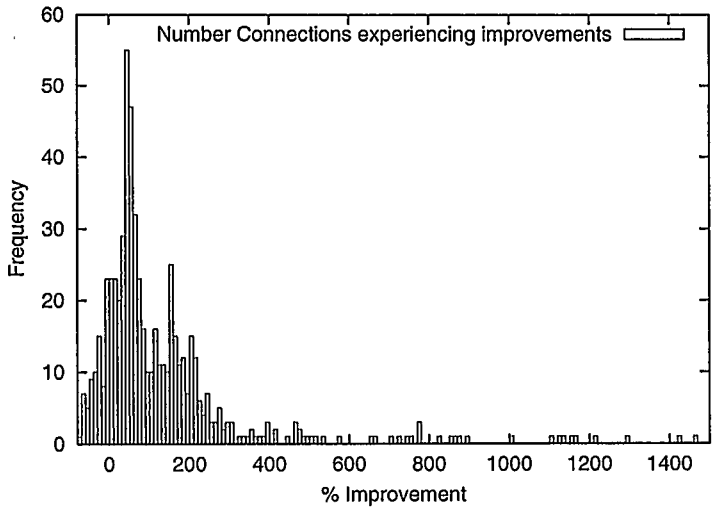

Figure 5.16: Histogram of # connections that receive % average improvement (50% threshold)

tain levels of improvement over all the connections. Superimposed on the histogram are the number of transfers performed by hosts in that category. In total there are 90 unidirectional host pairs. Of these, 43 had at least one transfer which had a predicted improvement using a split connection exceeding the 50% threshold.

The graph indicates that 12 of the 43 host pairs experienced reduced throughput on average when using split-TCP. However, these hosts only transferred 42 files in total for the duration of the experiment. Only three routes that experienced reduced throughput were selected more than three times. This implies that pathological cases, where the same split connection with poor performance was repeatedly chosen were rare.

Four host pairs received average improvements in excess of 200% on a regular basis. Two of these four hosts were not configured for high performance TCP transfers; maximum TCP buffer sizes were set too low. The split-TCP connections helped

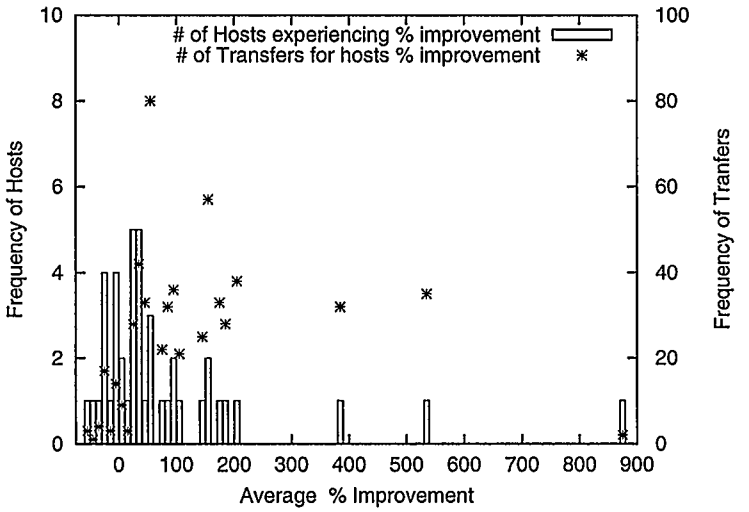

Figure 5.17: Histogram of number host pairs and number of transfers that receive average improvements in various ranges (50% threshold).

overcome this configuration error by reducing the RTT over which the small buffers applied. This type of configuration problem has been observed on many computers including HPC sites. However, not all significant gains are the result of poorly configured hosts. The 380% average improvement of the one host pair was the result of isolating a bottleneck router on a very low latency connection. The host pair with 538% average improvement experienced gains due to a combination of isolating the bottleneck and overcoming poorly configured maximum buffer sizes.

The host pair that experienced an improvement of 870% actually predicted an average improvement of 1600% during a narrow time period that only included 2 consecutive sample periods. The improvement occured only during the earlier of the two sample periods in which the split connection experienced an improvement of aproximately 1700%. There is no reason for this type of improvement between the host pairs in question that would remain over a period of time such as a misconfig-

uration. The latency between the two computers is low and the TCP buffers were configured correctly. An examination of the GridFTP logs reveals that one of the hosts went down. When the host came back up there was very poor connectivity between the two for a period of time. The connectivity had improved by the second test period. This indicates that the reason for the improvement was a transient network condition and the GridFTP Overlay network was capable of detecting and circumventing it by rerouting the traffic through a nearby proxy.

5.4 Summary

Both the emulation and wide area network experiments indicated that splitting a TCP connection can improve throughput in a variety of network conditions. This section summarizes the results from both sets of experiments.

The emulation experiments demonstrate that the GridFTP proxy servers using the split-DSI module are capable of improving throughput and link utilization across a variety of network environments. Although changes in the latency and background link utilization have the expected effect on throughput, increasing the MTU does not always improve throughput in any significant manner. A small improvement in throughput was observed by separating disk and network I/O. Results also demonstrated that although the bottleneck link can be very indicative of the throughput that will be achieved, significant overhead can be encountered due to ACK compression occurring on the interface. Ways to anticipate or avoid this overhead could dramatically improve performance. It is also important to note that both the overhead and the improvement from separating network and disk I/O may not occur on

different operating systems. Different networking hardware and scheduling policies in the operating system may change packet queueing behavior in a manner that leads to more, less or no ACK-compression.

The results of the wide area network experiments demonstrate the utility of the overlay network. Results indicate that good decisions on where to split connections can be made by the Split Choice Service with very sporadic data transfer information that is available from GridFTP transfer logs. The specific prediction metric used does significantly affect the results. Throughput improvements in excess of 500% were achieved on some transfers and an average throughput improvement of 130% for all split-TCP connections was attained with a minium predicted improvement threshold of 50%. The GridFTP overlay network also showed that it could provide a mechanism to overcome transient network conditions.

Chapter 6

Conclusion

The high performance computing (HPC) community is adopting the Grid computing paradigm. Grid computing, the presentation of computation, storage, networking, and instrumentation resources as a set of services, creates an infrastructure that enables the quick development and deployment of scientific applications. This infrastructure provides the ability to federate resources to solve larger scientific problems and supports new opportunities for collaboration, both of which increase demands on the networking infrastructure.

The physical networking infrastructure is becoming progressively more heterogeneous. Parts of the networking infrastructure are becoming exceedingly fast and there is increasing interest in scheduable quality of service guaranteed bandwidth. Other parts of this networking infrastructure, particularly that for networked instrumentation and sensors, consists of wireless network mediums which are often much slower and more error prone.

TCP, the most commonly used protocol for bulk data transmission often fails to meet the networking demands of data-intensive, distributed scientific applications. TCP fails to effectively utilize networks in which connections have high bandwidth delay products because recovery from errors take too long. TCP has poor utilization in heterogeneous network environments because bit errors are incorrectly interpreted as congestion and the window size reduced when it is unnecessary. The poor utilization experienced with TCP becomes worse as the latency of the connection increases.

As scientific applications make use of more distributed resources, connection latency will become more of a problem. It has been recognized for some time, that as networks become faster, and more heterogeneous in their characteristics, TCP's ability to utilize the network will become worse.

In order to overcome the poor utilization experienced using TCP, splitting a TCP connection has been suggested. The use of proxies to split TCP connections into several smaller segments with buffers between them overcomes TCP's poor utilization in a variety of environments. The majority of schemes developed for splitting a TCP connection were designed for specific environments. Using proxies to split all TCP connections is not practical because it would require too much memory in the routers and assumes that all TCP connections favour bulk transport capacity over other connection attributes such as low latency or reduced jitter. This means a split-TCP architecture requires mechanisms to discover the capacity of links and location of proxies, as well as, security mechanisms to manage access.

This thesis presented a set of components that enable easy deployment of overlay networks that make use of split-TCP connections to improve GridFTP transfer performance. The components include an extension to the Globus Toolkit v4 GridFTP server that supports split TCP connections, a service to estimate bulk transfer capacity and a service to determine if and where to split a connection. Together, these components provide the resource discovery, resource provisioning, authentication and authorization services required to implement split-TCP within data transfers. By using a widely adopted data transfer standard, the GridFTP overlay network can be easily integrated with existing distributed applications. A proxy server can be deployed by compiling a module for the widely deployed GT4 GridFTP server. Use

of the proxies does not require specialized clients. The proxies have been successfully used with a variety of GridFTP clients that were not developed with the us proxies in mind.

Emulation results indicated that the proxy server developed for this thesis has the potential to increase throughput for GridFTP transfers in a wide variety network conditions. Use of the proxy servers was also shown to decrease variance in transfer time of GridFTP transfers on both an absolute basis and as a percentage of the mean. Emulation results also revealed that using a host as a proxy for data transfers may result in ACK-compression on some systems. This source of overhead had not previously been discussed in the literature.

Results from a deployment of a GridFTP overlay network demonstrate significant performance improvement despite using very intermittent, passive throughput observations to determine the routing of the split connections. Throughput improvements in excess of 500% were achieved on some transfers and an average throughput improvement of 130% for all split-TCP connections was attainable when a 50 % threshold was used. The GridFTP overlay network also demonstrated an ability to overcome some transient, adverse network conditions.

At some point in the future, other protocols that are more effective in transferring large amounts of data in wide area networks could be used in place of TCP. The Globus Toolkit v4 GridFTP server was developed with an eXtensible Input/Output (XIO) [3] library that makes it possible to replace underlying protocol modules for this reason. However, the replacement of TCP with protocols that more effectively utilize bandwidth is taking a considerable amount of time. The proxy mechanisms developed in this thesis are a useful way to overcome TCP's inefficiencies while

other protocols are being developed tested and deployed. When new protocols are developed the proxy mechanisms developed here can still be used to make effective use of on demand bandwidth technologies such as scheduable light paths. The proxies provide a convenient method of routing the traffic to the ingress of the light path. This solves the first mile problem of getting the data routed to the lightpath. For security reasons, it is difficult to dynamically change the routing for a specific stream at the operating system level. The bandwidth estimation techniques used in this work could also be used to achieve efficient allocation of scheduable light paths, by ensuring only the bandwidth available over the first mile is reserved for a transfer. A considerable amount of work remains to be done in this area.

There is a substantial amount of future work related to proxy selection that remains to be explored. The emulation and wide area network experiments presented in this thesis both indicated there can be significant overhead. Improved ways of predicting additional overhead from split-points could dramatically improve the performance of split-TCP. Making use of previous performance in choosing proxies is one possibility. The possibility of dynamically changing the proxy being used during a transfer could also be examined. The proxy being used could be changed in response to changes in the network conditions or observing that actual throughput is not matching predicted throughput.

The implications of having a large number of split-TCP connections active concurrently needs to be explored. Large numbers of users using proxies may change the performance of the GridFTP overlay network. The implications the use of proxies has on the fairness of bandwidth allocation between streams also needs to be explored. Although the individual TCP connection will be fair, use of a proxy by

a user in certain situations may increase fairness issues experienced by a user not using proxies. In addition, more scalable algorithms to choose split points need to be developed. Making use of split-TCP in conjunction with parallel streams needs to be examined. Both split-TCP and parallel streams improve performance but it is not clear how complimentary the methods are.

Bibliography

[1] Internet protocol. RFC 791, Information Sciences Institute, 1981.

[2] W. Allcock, J. Bester, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke. GridFTP: protocol extensions to FTP for the Grid. Proposed Recommendation GFD-R-P.020, Global Grid Forum, April 2003.

[3] W. Allcock, J. Bresnahan, R. Kettimuthu, and J. Link. The globus extensible input/output system (xio): A protocol independent io system for the grid,. In *19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[4] M. Allman, S. Ostermann, and H. Kruse. Data transfer efficiency over satellite circuits using a multi-socket extension to the file transfer protocol. In *In Proceedings of the ACTS Results Conference*, 1995.

[5] M. Allman and V. Paxson. TCP congestion control. RFC 2581, Network Working Group, 1999.

[6] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 131–145, New York, NY, USA, 2001. ACM Press.

[7] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, Dec. 2004.

[8] A. Bakre and B. Badrinath. I-tcp: indirect TCP for mobile hosts. In *Proceedings*

of the International Conference on Distributed Computing Systems. IEEE press, May 1995.

[9] H. Balakrishnan, R. H. Katz, and V. N. Padmanbhan. The effects of asymmetry on TCP performance. *Mob. Netw. Appl.*, 4(3):219–241, 1999.

[10] H. Balakrishnan, S. Seshan, and R. H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1(4):469–481, 1995.

[11] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz. Analyzing stability in wide-area network performance. In *SIGMETRICS '97: Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 2–12, New York, NY, USA, 1997. ACM Press.

[12] V. Bharadwaj. Improving TCP performance over high-bandwidth geostationary satellite links. In *University of Maryland Technical Reports.* Institute for Systems Research, University of Maryland, 1999.

[13] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance enhancing proxies intended to mitigate link-related degradations. RFC 3135, The Internet Society, 2001.

[14] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the internet, 1998.

[15] R. Bradford, R. Simmonds, and B. Unger. A parallel discrete event IP network emulator. In *Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 315–322, 2000.

[16] L. Brakmo and L. Peterson. TCP vegas: end to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1490, Oct 1995.

[17] K. Brown and S. Singh. M-TCP: TCP for mobile cellular networks. *Computer Communication Review*, 27(5):19–43, 1997.

[18] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1742–1751, 2000.

[19] J. Cheng, D. Wei, and S. Low. Fast tcp: motivation, architecture, algorithms, performance. In *Proceedings of INFOCOM 2004 , Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2490– 2501, 2004.

[20] R. Cohen and S. Ramanathan. Using proxies to enhance TCP performance of hybird fiber coaxial networks. In *HP Labs Technical Reports*. HP Labs, 1997.

[21] J. Crowcroft and P. Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. *SIGCOMM Comput. Commun. Rev.*, 28(3):53–69, 1998.

[22] D. Doval and D. O'Mahony. Overlay networks: A scalable alternative for p2p. *IEEE Internet Computing*, 7(4):79–82, 2003.

[23] GT4.0 GridFTP users guide. http://globus.org/toolkit/docs/4.0/data/gridftp/user-index.html, 2006.

[24] R. Durst, G. Miller, and J. Travis. Tcp extensions for space communications. *Wirel. Netw.*, 3(5):389–403, 1997.

[25] M. L. N. Ehsan. Modeling TCP performance with proxies. In *International Workshop on Wired/Wireless Communications*, June 2002.

[26] S. T. et. al. Open grid services infrastructure (ogsi) version 1.0. global grid forum, proposed recomendation gfd-r-p.15. http://www.ggf.org/documents/GFD.15.pdf, 2003.

[27] A. Falk, T. Faber, J. Bannister, A. Chien, R. Grossman, and J. Leigh. Transport protocols for high performance. *Commun. ACM*, 46(11):42–49, 2003.

[28] A. Falk, T. Faber, J. Bannister, A. Chien, R. Grossman, and J. Leigh. Transport protocols for high performance. *Commun. ACM*, 46(11):42–49, 2003.

[29] K. Fall and S. Floyd. Simulation-based comparisons of tahoe, reno and sack tcp. *SIGCOMM Comput. Commun. Rev.*, 26(3):5–21, 1996.

[30] W. Feng and P. Tinnakornsrisuphap. The failure of TCP in high-performance computational grids. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 37, Washington, DC, USA, 2000. IEEE Computer Society.

[31] S. Floyd. Highspeed tcp for large congestion windows. RFC 3649, ICSI, 2003.

[32] M. Fomenkov, K. Keys, D. Moore, and K. Claffy. Longitudinal study of internet traffic in 1998-2003. In *WISICT '04: Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6. Trinity College Dublin, 2004.

[33] I. Foster. Globus toolkit version 4: Software for service-oriented systems. *IFIP International Conference on Network and Parallel Computing*, pages 2–13, 2005.

[34] I. Foster. The Holy Grail: Industry Wide System Managment at Last. http://globus.org/wsrf/convergence.php, 2005.

[35] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, New York, NY, USA, 1998. ACM Press.

[36] P. Gevros, F. Risso, and P. Kirstein. Analysis of a method of differential TCP server. In *Proceedings of Global Telecommunications Conference*, volume 3, pages 1699–1708, 1999.

[37] http://www.ggf.org, July 2006.

[38] Y. Gu and R. L. Grossman. Sabul: A transport protocol for grid computing. *Journal of Grid Computing*, pages 377–386, 2004.

[39] Y. H. H. Pucha. Overlay tcp: Ending end-to-end transport for higher throughput. In *Proceedings of SIGCOMM (as poster paper)*, 2005.

[40] G. Hasegawa, M. Murata, and H. Miyahara. Fairness and stability of congestion control mechanisms of TCP. In *INFOCOM '99 Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 3, pages 1329–1336. IEEE, Mar 1999.

[41] E. He, J. Leigh, O. Yu, and T. Defanti. Reliable blast udp : predictable high performance bulk data transfer. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 317– 324, 2002.

[42] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer TCP throughput. *SIGCOMM Comput. Commun. Rev.*, 35(4):145–156, 2005.

[43] J. Hollingsworth and B. Tierney. Instrumentation and monitoring. In I. Foster and K. Kesselman, editors, *The Grid 2: Blueprint for a New Computing Paradigm*, pages 391–429. Morgan Kaufman, 2004.

[44] B. Hubert, T. Graf, G. Maxwell, M. Oosterhout, P. Schroeder, J. Spaans, and P. Larroy. Linux advanced routing and traffic control howto. Technical report, lartc, 2005.

[45] ISO/IEC. Iso open systems interconnection (OSI) - basic reference model-: The basic modil. Technical report, Internation Organization for Standardaziation, International Electrotechnical Commision, 1994.

[46] T. Ito, H. Ohsaki, and M. Imase. On parameter tuning of data transfer protocol GridFTP for wide-area grid computing. In *2nd International Conference on Broadband Networks*, pages 415–421, 2005.

[47] V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, 1988.

[48] V. Jacobson, R. Braden, and D. Borman. Tcp extensions for high performance. RFC 1323, The Internet Society, 1992.

[49] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics and relation with TCP throughput. *ACM SIGCOMM*, 2002.

[50] X. Jianxuan, M. A. Labrador, and M. Guizani. Performance evaluation of TCP over optical channels and heterogeneous networks. *Cluster Computing*, 7(3):225–238, 2004.

[51] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Improving TCP throughput over two-way asymmetric links: analysis and solutions. In *SIGMETRICS '98/PERFORMANCE '98: Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 78–89, New York, NY, USA, 1998. ACM Press.

[52] D. Katabi. *Decoupling Congestion Control and Bandwidth Allocation Policy.* PhD thesis, Massachusetts Institue of Technology, 2003.

[53] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 89–102, New York, NY, USA, 2002. ACM Press.

[54] A. V. L. Kalampoukas and K. K. Ramakrishnan. Two-way TCP traffic over ATM: Effects and analysis. In *in Proc. of IEEE INFOCOM97*, April 1997.

[55] T. Lakshman and U. M. U. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEE/ACM Transactions on Network*, 5:336–350, June 1997.

[56] R. Madduri, C. Hood, and W. Allcock. Reliable file transfer in grid environments. In *Proceedings of 27th Annual IEEE Conference on Local Computer Networks*, pages 737– 738. IEEE press, 2002.

[57] I. Maki, G. Hasegawa, M. Murata, and T. Murase. Performance analysis and improvement of TCP proxy mechanism in TCP overlay networks. *Proceedings of the IEEE International Conference on Communications*, 1:184–190, 2005.

[58] M. Mathis and M. Aliman. A framework for defining empirical bulk transfer capacity metrics. RFC 3148, Network Working Group, 2001.

[59] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. RFC 2018, Network Working Group, 1996.

[60] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997.

[61] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communication Review*, 27(3):67–82, 1997.

[62] A. Medina, M. Allman, and S. Floyd. Measuring the evolution of transport protocols in the internet. *SIGCOMM Comput. Commun. Rev.*, 35(2):37–52, 2005.

[63] J. Mo, R. La, V. Anantharam, and J. Walrand. "analysis and comparison of TCP reno and vegas. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1556–1563, 1999.

[64] A. Networks. Extended frame sizes for next generation ethernet. Alteon White Paper.

[65] R. Nitzan and B. Tierney. Experiences with tcp/ip over an atm oc12 wan. In *Gigabit workshop INFOCOMM99*, 1999.

[66] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[67] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, 2000.

[68] V. Paxson. End-to-end routing behavior in the internet. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 25–38, New York, NY, USA, 1996. ACM Press.

[69] V. Paxson. End-to-end internet packet dynamics. In *SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 139–152, New York, NY, USA, 1997. ACM Press.

[70] J. Postel. User datagram protocol, August 1980.

[71] J. Postel. Transmission control protocol DARPA internet protocol control specification, September 1981.

[72] G. Raina, D. Towsley, and D. Wischik. Part ii: control theory for buffer sizing. *SIGCOMM Comput. Commun. Rev.*, 35(3):79–82, 2005.

[73] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ecn) to ip. RFC 3628, The Internet Society, 2001.

[74] P. Rizk, C. Kiddle, and R. Simmonds. Improving gridftp performance with Split-TCP connections. In *1st IEEE Internationl Conference on e-Science and Grid Computing*, 2005.

[75] P. Rizk, C. Kiddle, and R. Simmonds. A GridFTP overlay network service. In *7th IEEE/ACM Internationa Conference on Grid Computing*, 2006.

[76] J. H. Salim and U. Ahmed. Performance evaluation of explicit congestion notification (ecn) in ip networks, 2000.

[77] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions in Computer Systems*, pages 277–288, 1984.

[78] S. Schenker, L. Zhang, and D. D. Clark. Some observations on the dynamics of a congestion control algorithm. *SIGCOMM Comput. Commun. Rev.*, 20(5):30–39, 1990.

[79] J. Schopf, M.D'Arcy, N. Miller, L. Pearlman, and I. F. C. Kesselman. Monitoring and discovery in a webservices framework: Functionality and performance of the globus toolkit's mds4. Technical Report ANL/MCS-P1248-04-5, Globus, 2005.

[80] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: The case for application-level network striping for data intensive applications using hish speed wide area networks. In *Supercomputing*, 2000.

[81] R. Stevens. *The Protocols (TCP/IP Illustrated, Volume 1).* Addison-Wesley Professional, December 1993.

[82] A. Sundararaj and D. Duchamp. Technical report: Analytical characterization of the throughputof a split TCP connection, 2003.

[83] M. Swany. Improving throughput for grid applications with network logistics. In *Proceedings of the ACM/IEEE Conference*, pages 23–31, 2004.

[84] M. Swany and R. Wolski. Data logistics in network computing: The logistical session layer. In *Proceedings of the IEEE International Symposium on Network Computing and Applications*, pages 174 – 185, 2001.

[85] S. Vanichpun and F. Wu-chun. On the transient behavior of TCP vegas.

In *Eleventh International Conference on Computer Communications and Networks*, pages 504–508, October 2002.

[86] S. Vazhkudai and J. Schopf. Predicting sporadic grid data transfers. In *11th IEEE International Symposium on High Performance Distributed Computing*, 2002.

[87] S. Vazhkudai, J. Schopf, and I. Foster. Predicting performance of wide area data transfers. In *Proceedings of the 16th Int'l Parallel and Distributed Processing Symposium*, 2002.

[88] R. Wilder, K. K. Ramakrishnan, and A. Mankin. Dynamics of congestion control and avoidance of two-way traffic in an osi testbed. *SIGCOMM Comput. Commun. Rev.*, 21(2):43–58, 1991.

[89] C. Williamson and Q. Wu. A case for context-aware tcp/ip. *SIGMETRICS Perform. Eval. Rev.*, 29(4):11–23, 2002.

[90] D. Wischik and N. McKeown. Part i: buffer sizes for core routers. *SIGCOMM Computer Communication Review.*, 35(3):75–78, 2005.

[91] R. Wolski, N. Spring, and C. Peterson. Implementing a performance forecasting system for metacomputing: the network weather service. In *Supercomputing '97: Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–19, New York, NY, USA, 1997. ACM Press.

[92] X. Wu and A. Chien. Evaluation of rate-based transport protocols for lambdagrids. In *Proceedings of 13th IEEE International Symposium on High perfor-*

mance Distributed Computing, pages 87–96, 2004.

[93] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(4):133–147, 1991.

Appendix A

Statement of published work

During the production of this thesis, two papers where published. The emulation results in this paper are derivative of similiar work published in:

P. Rizk, C. Kiddle, and R. Simmonds. Improving GridFTP performance with split TCP connections. In *1st IEEE International Conference on e-Science and Grid Computing*, 2005.

The author of this thesis developed the testing framework, performed the experiments and data analysis for the work in the paper. The writing tasks in this paper were shared among the authors.

A subset of the wide area network results in the thesis are published in:

P. Rizk, C. Kiddle, and R. Simmonds. A GridFTP overlay network service. In *7th IEEE/ACM International Conference on Grid Computing*, 2006.

The software development of the system, testing framework, data analysis and writing for this paper were done by the author of this thesis. The second and third authors provided guidance, dealt with administrative issues related to the multiple institutions involved, and assisted in editing the paper.

All the software development, testing framework, data analysis and writing for this thesis was done by the author.