A GridFTP Overlay Network Service

Philip Rizk [1], Cameron Kiddle [2], Rob Simmonds [3]

Department of Computer Science, University of Calgary
Calgary, Alberta, Canada
[1]rizkp@cpsc.ucalgary.ca
[2]kiddlec@cpsc.ucalgary.ca
[3]simmonds@cpsc.ucalgary.ca

***Abstract*— TCP is widely deployed and is used by the current implementation of GridFTP for data transfers within grid environments. Due to TCP dynamics, throughput of data transfers can be much lower than the available bandwidth in wide area networks. Splitting a TCP connection into two or more segments has been shown to improve throughput in this setting.**

This paper presents a set of components to 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. Results demonstrate significant performance improvement despite using very intermittent, passive throughput observations to determine the routing of the split connections.

I. Introduction

Grid computing technologies allow researchers around the world to share data and resources across administrative domains. The widely distributed nature of these environments makes the performance of data transfer increasingly important. TCP, the most widely deployed protocol for data transfer, suffers from significant performance problems in heterogeneous, wide area networks.

Tools using the GridFTP protocol [1] provide significant improvement in file transfer performance over other tools such as *ftp* or *scp* due to the ability to fine tune TCP buffer sizes and allow parallel streams. Still, the performance of a transfer will be limited by the slowest link in the network, and the time required to recover from errors in TCP increases with the round trip time (RTT) of the connection. These effects often result in underutilization of the network particularly in connections with high latency and connections where one segment has high loss for reasons other than congestion such as a high physical bit error rate.

Splitting a TCP connection into several segments has been suggested as a way to overcome some of the problems with TCP [2], [3], [4], [5], [6]. Emulation studies [7] have indicated that even when using large buffer sizes, GridFTP can benefit from split-TCP connections. This paper presents a practical implementation of a GridFTP overlay network that makes use of split-TCP connections to improve GridFTP transfer performance.

The GridFTP overlay network services presented in this paper have a very low deployment expense as only severs that act as proxies need to be modified. It is not necessary to modify GridFTP clients and servers at the edge of the overlay network. Experimental results demonstrate that throughput data available from GridFTP logs on very intermittently used servers can be used to arrive at effective split points.

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) [8] library that makes it possible to replace underlying protocol modules. Enabling split connections may still be of benefit to these future protocols as it provides the capability of overlay routings. These mechanisms support the use of scheduled network resources such as dynamically allocated light paths by solving the problem of routing to the light path.

The rest of this paper is organized as follows. Section II discusses related work. The components of a GridFTP overlay network are presented in section III. Section IV describes the experimental methodology and environment used to evaluate the overlay network. Section V presents performance results for the test implementation. A summary and future work is given in Section VI.

II. Related Work

A considerable amount of literature exists that discusses the advantages of split-TCP connections, overlay networks and improving the performance of file transfers.

GridFTP and PSockets [9] make use of parallel TCP connections at the application layer. Using multiple TCP streams effectively increases the rate at which TCP recovers from errors, reduces the rate at which it slows throughput in response to errors and increases the effective TCP-buffers space available. The two potential problems with this approach are the difficulty in determining the optimal number of streams to use and transfers getting an unfair share of the network. A split-TCP connection does not suffer from fairness issues because a single normal TCP connection is used on each segment. Parallel streams can still be very useful in known environments and can also compliment split-TCP approaches.

There has been a considerable amount of work into split-TCP approaches aimed at improving the performance of TCP. Most of these approaches provide a solution to poor TCP performance for a specific type of network such as wireless [2], [3], [4], satellite [5], [6] and Hybrid Fiber Coaxial (HFC)

networks [10]. These types of networks can in general be characterized by high latency and packet loss unrelated to congestion on a portion of the network. In all cases the reduced RTT results in a quicker recovery time from losses. The effect is a result of the fact that TCP recovers from errors at a rate in proportion to RTT [11].

There has been considerable work on modelling TCP performance. Models of TCP performance on both single [11], [12] and split connections [13], [14], [15] can be useful in inferring where splitting a connection can be beneficial. Previous work has demonstrated that errors in inputs or the assumptions of the model can have severe impacts on the ability of these models to accurately predict throughput [16], [17], [18], [7]. The main value in analytical models is insight into performance of TCP in specific environments. For example, [15] demonstrates that split-TCP connections require larger buffers.

The most closely related work to the work presented in this paper is the Logistical Session Layer (LSL) [19]. LSL splits TCP connections at strategic points based on TCP performance as reported by Network Weather Service (NWS) [20]. The LSL splits TCP connections when it observes multiple TCP connections in series will perform better than an individual connection. LSL is the first work to our knowledge that uses previous TCP performance on any link to arrive at split points for any network.

The approach presented in this paper differs in its source of network information and underlying protocol. Intermittent passive data is used to arrive at throughput information that includes disk access time, while LSL uses active probes of NWS. As NWS is a generic forecasting tool, it is possible to add measures that include more than network bandwidth. The GridFTP overlay network components also entail little deployment expense. Any GridFTP site can function as a split server by adding a module to the library used by the GridFTP server. Endpoints of a transfer (clients and servers) need no modification at all if passed URLs with proxy information already in them. The use of GridFTP also leverages the Grid Security Infrastructure (GSI) security protocol which allows fine grained control of the use of proxy servers.

Generalized TCP overlay networks [21], [15] have been proposed and tested primarily for the purposes of Web traffic. Overlay TCP [21] uses RTT exclusively while [15] uses both RTT and packet loss. Both proposals differ from the approach presented in this paper by 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 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 and skew performance predictions by orders of magnitude [18]. These systems also violate end-to-end TCP semantics.

Kangaroo is a wide area data movement system that hides network latency using memory and disk buffers [22]. The memory and disk buffers can be chained together across a wide area network. This has the effect of improving the performance in the same way splitting a GridFTP connection does. However, the performance improvements obtained by using the Kangaroo system were primarily from overlapping the CPU and I/O tasks of an application.

Resilient Overlay Network (RON) [23] is an application level overlay that exists on top of an existing network [23]. While the main goal of RON is to address network outages, it also improves throughput by rerouting packets using formula based prediction based on the simple throughput model developed in [24]. The RON architecture will capture any performance difference to be gained by rerouting a TCP connection over links with lower RTT or packet loss rates but does not split the TCP connection. This improves TCP performance but will not capture all of the gains to be made by reducing RTTs and recovering from losses locally that splitting the TCP connection will.

III. GridFTP Overlay Network Components

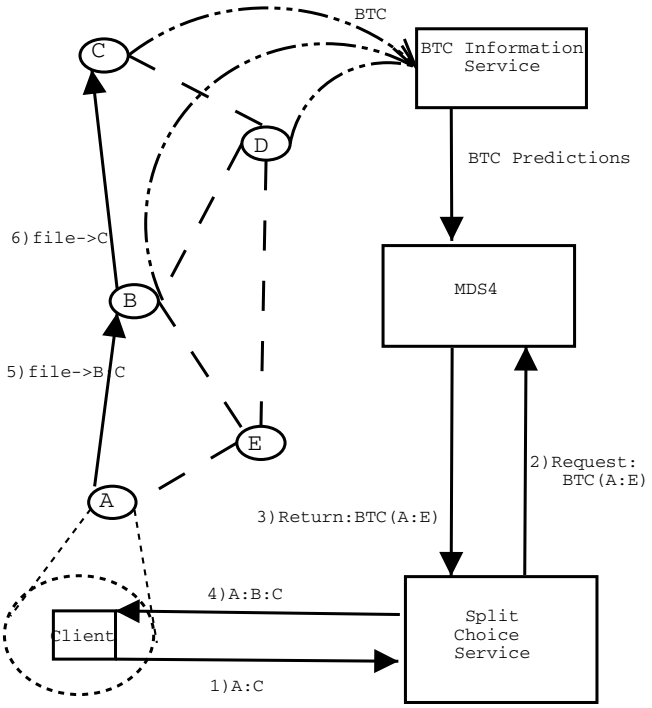

Fig. 1. GridFTP Overlay Network Architecture

A GridFTP overlay network consists of a group of GT4 servers that include a module to support split connections, a Split-Choice Service (SCS) for choosing the number and location split points, and a monitoring framework that utilizes GridFTP logs to make predictions of GridFTP transfer performance.

Figure 1 depicts how a file transfer client would use the overlay network to transfer a file. The client first queries the SCS to determine a split connection giving a source A and a destination C. The SCS then queries an MDS4 server [25] , an information service provider distributed with the Globus Toolkit, for recent throughput information. This information

is populated on an ongoing basis by a collector of results from normal data transfers. 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. Routing information of the split-TCP connections is included in the URL passed to the split server.

A. GT4 GridFTP server

The GridFTP protocol is defined by the Global Grid Forum recommendation GDF.020, RFC959, RFC2228, RFC2398. It is an extension of the FTP protocol, providing features useful for performing high performance transfers within heterogeneous, high latency networks often found in Grid environments. These features include but are not limited to the Grid Security Infrastructure (GSI), negotiated buffer sizes, and support for parallel and partial transfers [1].The GT4 GridFTP server [26] is a GridFTP protocol compliant server distributed with the Globus Toolkit [27].

B. Split-DSI module

The GT4 GridFTP server provides hooks to write a Data Storage Interface (DSI) [28]. The DSI is an interface that 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 can by 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. This effectively splits the GridFTP connection into two or more TCP connections, buffering data between the two connections. 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.

The most significant criticism of split-TCP approaches to improving TCP performance is the violation of end-to-end semantics. Splitting GridFTP connections does not have this problem since GridFTP has its own application level acknowledgements. It is still a significant problem for a more generalized splitting of TCP connections. In addition to improved TCP throughput, the overlay network can be used to reroute packets over alternate paths. This is useful when using network infrastructures that treat network paths as scheduled objects. An examples of this is User Controlled Light-Paths (UCLP) [29]. UCLP allows users to dynamically control and provision optical networks across management domains. The aim is a system in which users can check on the availability of a light path for a transfer, and if available, route traffic through it. One of the challenges of doing this is routing the packets to the ingress of the light path. The Split-DSI module creates an overlay network to reroute the packets.

The Split-DSI module gets routing information from the URL it receives from the GridFTP client. This allows seamless integration with current clients. We have successfully used the split server with *globus-url-copy*, unmodified GridFTP servers (as endpoints) and the Reliable File Transfer Service (RFT) [30].

C. Bulk Transfer Capacity Information Service

The Bulk Transfer Capacity (BTC) information service provides a prediction of achievable 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 [31]. Achievable throughput is the 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 and packet loss. Achievable throughput is usually lower than available bandwidth but it was noted in [31] that achievable bandwidth may be greater than available bandwidth due to the effects of the new stream on other streams. In addition to network characteristics, throughput can also be affected by disk access speeds and the load on the computer.

BTC prediction can be based on either the underlying characteristics of a network such as RTT and loss rates, as well a model to predict throughput or on past experience of the applications. Previous work has described this distinction as formula based (FB) versus history based (HB) prediction [18].

Formula based prediction involves getting information such such as MTU, RTT, and loss rates. A TCP performance model uses this information to predict throughput. One difficulty in utilizing this approach is obtaining network measurements. The level of detail required for each link in the network is often not available. When tools are available, however, they tend to be less intrusive.

The TCP performance models can be sensitive to errors in the measurement of the underlying characteristics. It has been shown that because TCP performance predictions are based on measurements taken without the flow to be added, the added flow will increase actual RTT and loss; this often causes an overestimation of achievable throughput by orders of magnitude [18]. Performance models are often 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 such as when a connection is receive buffer limited [7].

History Based prediction techniques use previous experience, preferably of the application in question, to make throughput predictions. In contrast to FB techniques, they tend to be readily available assuming the application can be instrumented appropriately. Previous work has indicated that even relatively simple techniques such as maintaining an average or median throughput can be quite accurate [18], [17]. GridFTP throughput predictions based on GridFTP logs can be made even more accurate when used in conjunction with Network Weather Service (NWS) [16]. NWS has been found to be inaccurate on high bandwidth ($\geq 1Gbs$) links when used

alone [16], [17]. For the GridFTP overlay network service, predictions are not enhanced with NWS as one of the goals is to determine the minimum amount of information necessary for effectively splitting a connection.

The GridFTP overlay service uses a hierarchical BTC information service that reports a prediction of BTC based on various simple prediction techniques using only GridFTP transfer logs. This provides unobtrusive passive monitoring. The information is then aggregated and disseminated through MDS4 [25].

The BTC Information Service is similar to the system described in [17] which provides reasonable estimates of throughput with errors not exceeding 25% in most cases. It is not the intention of this paper to suggest new history based prediction techniques but to test old ones with our data set and utilize this work to select split points.

The median of a data set W is formally defined as the value $X_i \in W$ such that for all $X_j \in W$ where $j \neq i$ $1/2$ of the elements are such that $X_j \leq X_i$ and $1/2$ of of the elements are such that $X_i \geq X_j$. For the purposes of experiments for this paper, moving median was taken using last 5 measurements. Medians were chosen because they ignore outliers. Other linear predictors were tested but results of the experiments presented in Section V did not differ significantly.

Both disk to disk (D2D) and memory to memory (M2M) transfers were performed with the moving median predictor. D2D moving median reports disk to disk transfers which is what a GridFTP client normally performs. M2M moving median data reports an estimate of memory to memory GridFTP transfers by transferring from /dev/zero at the source to /dev/null at the destination and recording the amount of data transferred. This was done to get some sense of the degree to which disk performance is affecting GridFTP transfers in the test network. Ideally information of transfer times to and from particular volumes would be provided. This would add to the complexity of the system.

D. Split-Choice Service

The Split-Choice Service (SCS) is currently 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 any. 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 maximizes the minimum bulk transport capacity (BTC) along the way.

In [32] the creators of LSL recognized that this could be achieved by building a minimum spanning tree for each host. The authors use a modified greedy tree building algorithm. In addition to this, the authors partition many hosts into groups with similar bandwidth. This greatly simplifies the routing tree built and can minimize extraneous hops. This has a complexity of O(NlogN) for an implementation that keeps BTCs in sorted order.

SCS uses a brute force search. This was not overly expensive because of the small fixed number of split points used for the overlay topology described in section IV. This would be sufficient for many grid applications. If more scalability was required the LSL algorithm would be more appropriate. This 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 maximizing the minimum capacity will result in a bandwidth somewhat similar to the minimum capacity with marginal 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 others 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. Further study is required to determine the exact causes of, and how to predict, this behaviour.

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 users needs and level of control over the computers being used.

An aim is to provide split point selection as service that will examine both historical throughput and scheduable bandwidth such as dynamic lightpath allocation through UCLP that is available to the user. The SCS will then make bandwidth reservations (if necessary) and return the split point selections to the client. To maintain scalability it is likely that this service would be structured as a hierarchy of SCS servers.

IV. Experimental Methodology

This section describes the overlay network topology and methodology used to evaluate the performance of the system.

A. GridFTP Overlay Network Topology

The GridFTP overlay network topology used for experiments is shown in Figure 2. It is comprised of ten hosts including three hosts from the University of Calgary Grid Research Centre (GRC), five hosts from Westgrid, one host from 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 CCN Grid is an HP led grid computing collaboration that the GRC is also 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 CA*net4 in Canada and Abilene in the United states. The majority of hosts are high performance computing facilities.

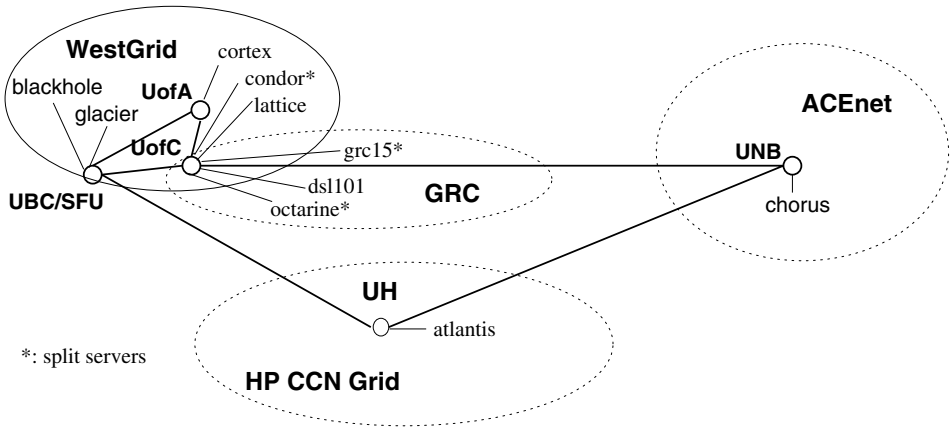

Fig. 2. GridFTP Overlay Network Topology

B. 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 modelled 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. This meant that the throughput for 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 32MB to 1024 MB. A transfer duration of 90 seconds was chosen because preliminary experiments determined this was sufficient to achieve steady state behaviour.

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 use of caching on disk systems means that what was intended to be a D2D transfer may have in fact been M2D , D2M or M2M. Tests were undertaken and the use of random files on some hosts was implemented in order minimize disk caching effects. Disk caching is only an issue when the disk, not the network is a bottleneck.

All transfers were performed using a GridFTP client built for the purposes of these experiments using the API distributed with the Globus Toolkit. The client reported performance information every time it was received from the destination server. Time measurements for throughput were taken using the performance plugin from the GridFTP libraries. Connection negotiation time was taken into account.

C. Analysis of split point selection

Possible splits for all host pairs were evaluated at random intervals with an exponentially distributed inter-arrival time with a mean of 6 hours. The possible splits were evaluated based on the D2D and M2M throughput predictions. On any connection in which a split connection had higher estimated bandwidth than the direct connection for either the M2M or D2D predictors, a split-connection and a control connection were performed one after the other. The control connection was a single connection from the source to the destination. If the M2M and D2D predictions chose different split points, both were attempted.

V. 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.

A. Accuracy of BTC Predictors

TABLE I
TABLE OF ACCURACY FOR BTC PREDICTORS

Metric	Average % error
D2D Median	21.05%
M2M Median	177.67%

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

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

Table I shows the average percent error of the predicted throughput for all of the actual disk to disk (D2D) control transfers for the two metrics being examined. D2D transfer estimates for this network have similar levels of accuracy as those found in [17]. 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

TABLE II
PERFORMANCE OF SPLIT-TCP CONNECTIONS

Number of Transfers	1372
Average Predicted % Improvement	62.32
Average Actual % Improvement	47.29
95% Conf. Radius (Actual Improvement)	6.94

indicates that disk I/O can have a large effect on GridFTP performance and attempts to predict throughput should take this into account.

B. Overall Performance of Split-TCP Connection

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

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

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 D2D moving median to decide where to split connections appears in Table II. A total of 6 datapoints were removed from the data set because the predicted improvement of these points was greater than 6 standard deviations from the mean; this distorted the predicted improvement average significantly. These points are included in the rest of the results. The average actual improvement was approximately 47%. Split-TCP connections were used whenever predicted improvement was greater than zero. Typically, a split-TCP conection would only be chosen if the predicted improvement is greater than a certain threshold. Figure 3 shows what the average improvement over single connections is for split-TCP transfers for different predicted improvement thresholds. The larger the thereshold used the greater the average performance improvment. For example, if a minimum predicted improvement of 50% is used, the average improvement achieved exceeded 125%.

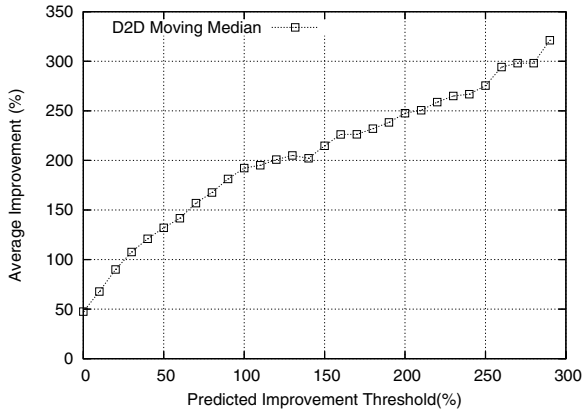

Fig. 3. Average improvement versus predicted improvement threshold.

Figure 4 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 single connections. In comparison, 35% 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.

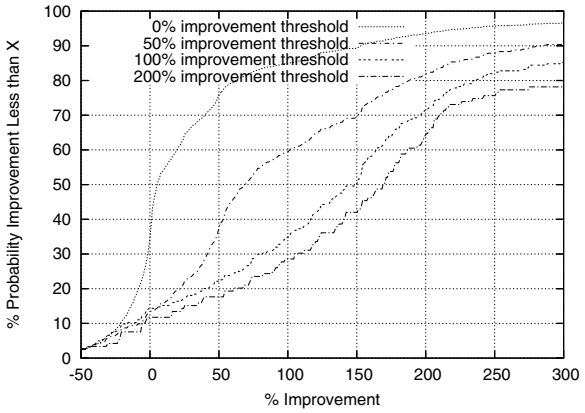

Fig. 4. CDF of split-TCP connection improvements for various prediction thresholds

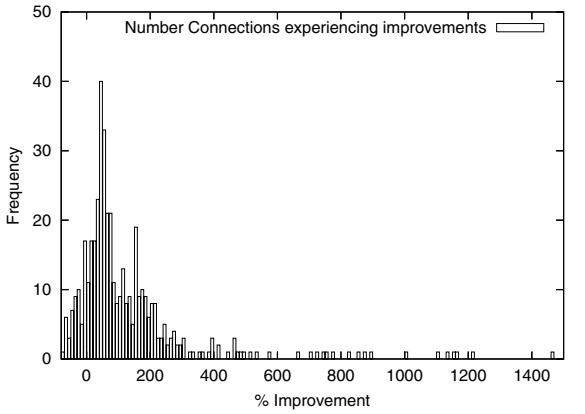

Fig. 5. Histogram of # connections that receive % average improvement (50% threshold)

Figure 5 shows a histogram of the number of connections that experienced various ranges of improvement where the minimum predicted improvement threshold is 50%. A majority of split connections had a 30% to 200% improvement in throughput. Only 13% of the connections experienced reduced throughput.

Figure 6 shows a histogram of the number of host-pairs that experienced certain 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, 40 had at least one transfer which had a predicted improvement using a split connection exceeding the 50% threshold.

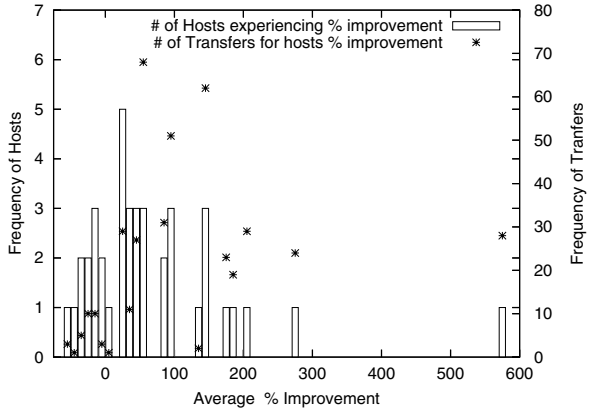

Fig. 6. Histogram of number host pairs and # of transfers that receive average improvements in various ranges (50% threshold).

The graph indicates that 11 of the 40 host pairs experienced reduced throughput on average when using split-TCP. However, these hosts only transferred 32 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.

Two of the three host pairs that received average improvements in excess of 200% were at least partly related to computers that were not configured for high performance TCP transfers; maximum TCP buffer sizes were set low. We have encountered this on many computers including HPC sites. Not all significant gains are the result of poorly configured hosts, however. The 272% 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 573% average improvement experienced gains due to a combination of isolating the bottleneck and overcoming poorly configured maximum buffer sizes.

VI. Conclusion and Future Work

This paper presented a set of components that enable the creation of an overlay network that uses split-TCP connections to improve GridFTP transfer performance. The components consist of a module that extends the GT4 GridFTP server to support the use of split-TCP connections, an information service that predicts throughput based on GridFTP logs, and a service to choose servers at which to split a connection. The overlay network is simple to deploy and can be used by unmodified GridFTP clients.

The results of experiments to show the utility of the overlay network indicate that good decisions on where to split connections can be made with very sporadic data transfer information. Throughput improvements in excess of 500% were achieved on some transfers and an average throughput improvement of 125% for all split-TCP connections was attained with a minium predicted improvement threshold of 50%.

There are many areas related to this work that need to be explored. Improved ways of predicting additional overhead from split-points must be developed. The implications of having a large number split-TCP connections active concurrently needs to be explored. 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.

In the future, better suited protocols could be used in place of TCP for wide area network data transfers. However, split connections may still be useful to isolate poorly performing segments of the network and reroute connections over specific segments. This ability to reroute packets may be particularly useful when utilizing bandwidth on demand networks.

Acknowledgements

This work is part of a CANARIE funded project and was performed at the Grid Research Centre (GRC) at the University of Calgary. The GRC receives funding from Hewlett Packard, Alberta Innovation and Science,Western Economic Diversification Canada. Additional funding is received from ASRA (Alberta Science and Research Authority) and NSERC (Natural Sciences and Engineering Research Council of Canada). We would also like to acknowledge the use of WestGrid, ACEnet and HP CCN Grid resources. WestGrid and ACEnet are funded by the Canadian Foundation for Innovation (CFI) and provincial funding bodies. HP CCN Grid is a collaborative project led by Hewlett Packard.

References

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

[2] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proceedings of the International Conference on Distributed Computing Systems*, 1995, pp. 136–143.

[3] B. R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz, "Handling mobile clients: A case for indirect interaction," in *Proceedings of the Fourth Workshop on Workstation Operating Systems*, 1993, pp. 91–97.

[4] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *Wireless Networks*, vol. 1, no. 4, pp. 469–481, 1995.

[5] T. R. Henderson and R. H. Katz, "Transport protocols for Internet-compatible satellite networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 2, pp. 326–344, 1999.

[6] M. Luglio, M. Y. Sanadidi, M. Gerla, and J. Stepanek, "On-board satellite "Split TCP" proxy," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 2, pp. 362–370, 2004.

[7] 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.

[8] 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.

[9] 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.

[10] R. Cohen and S. Ramanathan, "Using proxies to enhance TCP performance over hybrid fi ber coaxial networks," *Computer Communications*, vol. 20, no. 16, pp. 1502–1518, 1998.

[11] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.

[12] 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*, vol. 8, no. 2, pp. 133–145, 2000.

[13] N. Ehsan and M. Liu, "Modeling TCP performance with proxies," *Computer Communications*, vol. 27, no. 10, pp. 961–975, 2004.

[14] A. I. Sundararaj and D. Duchamp, "Analytical characterization of the throughput of a split TCP connection," Department of Computer Science, Stevens Institute of Technology, Technical Report 2003-04, 2003.

[15] 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*, vol. 1, pp. 184–190, 2005.

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

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

[18] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 145–156, 2005.

[19] M. Swany and R. Wolski, "The logistical session layer," in *Proceedings. 10th IEEE International Symposium on High Performance Distributed Computing*, 2001, pp. 443 – 444.

[20] R. Wolski, N. Spring, and C. Peterson, "Implementing a performance forecasting system for metacomputing: the network weather service," in *Proceedings of the ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM Press, 1997, pp. 1–19.

[21] Y. H. H. Pucha, "Overlay TCP: Ending end-to-end transport for higher throughput," in *Proceedings of SIGCOMM (as poster paper)*, 2005.

[22] D. Thain, J. Basney, S.-C. Son, and M. Livny, "The kangaroo approach to data movement on the grid," in *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, August 2001. [Online]. Available: citeseer.ist.psu.edu/thain01kangaroo.html

[23] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the ACM symposium on Operating Systems Principles*. New York, NY, USA: ACM Press, 2001, pp. 131–145.

[24] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 303–314, 1998. [Online]. Available: citeseer.csail.mit.edu/padhye98modeling.html

[25] 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," Globus, Tech. Rep. ANL/MCS-P1248-04-5, 2005.

[26] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus striped GridFTP framework and server," in *Proceedings of Supercomputing (SC05)*, 2005.

[27] I. Foster, "Globus Toolkit version 4: Software for service-oriented systems," in *IFIP International Conference on Network and Parallel Computing*, 2005, pp. 2–13.

[28] "GT4.0 GridFTP users guide," http://globus.org/toolkit/docs/4.0/data/gridftp/user-index.html, 2005.

[29] J. Wu, S. Campbell, J. Savoie, and H. Zhang, "User-managed end-to-end light path provisioning over CA*net4," in *Proceedings of the National Fiber Optic Engineers Conference*, 2003.

[30] R. Madduri, C. Hood, and W. Allcock, "Reliable fi le transfer in grid environments." IEEE press, 2002, pp. 737– 738.

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

[32] M. Swany, "Improving throughput for grid applications with network logistics," 2004, pp. 23–31.