On Parameter Tuning of Data Transfer Protocol GridFTP for Wide-Area Grid Computing

Takeshi Ito, Hiroyuki Ohsaki and Makoto Imase
Graduate School of Information Science and Technology, Osaka University
1-5, Yamadaoka, Suita, Osaka, 560-8531, Japan
E-mail: {t-itou, oosaki, imase}@ist.osaka-u.ac.jp

***Abstract*— In wide-area Grid computing, geographically distributed computational resources are connected for enabling efficient and large-scale scientific/engineering computations. In the wide-area Grid computing, a data transfer protocol called *GridFTP* has been commonly used for large file transfers. GridFTP has the following features for solving problems of the existing TCP. First, for accelerating the start-up in TCP's slow start phase and achieving high throughput in TCP's congestion avoidance phase, multiple TCP connections can be established in parallel. Second, according to the bandwidth-delay product of a network, the TCP socket buffer size can be negotiated between GridFTP server and client. However, in the literature, sufficient investigation has not been performed either on the optimal number of TCP connections or the optimal TCP socket buffer size. In this paper, we therefore quantitatively investigate the optimal parameter configuration of GridFTP in terms of the number of TCP connections and the TCP socket buffer size. We first derive performance metrics of GridFTP in steady state (i.e., goodput and packet loss probability). We then derive the optimal parameter configuration for GridFTP and quantitatively show performance limitations of GridFTP through several numerical examples. We also demonstrate validity of our approximate analysis by comparing simulation results with analytic ones.**

I. Introduction

Transmission Control Protocol (TCP) has been widely used as a transport-level communication protocol in the Internet [1]. GridFTP has been designed for utilizing TCP as its transport-level communication protocol [2]. However, TCP is a rather old communication protocol that was designed in the 1970s. Several problems have been reported regarding TCP such as its inability to support the rapidly increasing speeds of recent networks.

As an example, the current TCP Reno (TCP version Reno) cannot detect congestion in a network until packet loss occurs, so a large number of packets are lost. With the faster speeds of networks and larger buffer sizes of routers in a network, the amount of packets lost mushrooms and TCP throughput deteriorates significantly. To resolve existing TCP problems, GridFTP has features such as establishing multiple TCP connections in parallel to accelerate start-up in the TCP slow start phase and negotiating the TCP socket buffer size between the GridFTP server and client according to the bandwidth-delay product of a network [2].

However, the effectiveness of these features has not been fully investigated. In other words, optimal configurations for the number of parallel TCP connections and TCP socket buffer size have not been investigated. There have been several related works on TCP socket buffer size and parallel TCP connections [3-9].

In [3, 4], automatic tuning mechanisms of TCP socket buffer have been proposed. However, both approaches require some modifications to a socket API and/or a TCP protocol stack. In Grid computing, heterogeneous computing resources are integrated. Hence, such modifications to operating systems are unrealistic; i.e., optimization of TCP socket buffer size should not rely on any information obtained from other than Grid middleware. In [5], an extension to GridFTP protocol for automatically negotiating TCP socket buffer size has been proposed. However, the proposed mechanism is simple and not optimal; i.e., it simply allocates twice of the BDP (bandwidth-delay product) for each TCP connection. For efficient memory allocation to TCP socket buffer, more intelligent mechanism must be incorporated.

On the contrary, effectiveness of parallel TCP connections has been studied by many researchers [6-9]. In [6-9], performance of parallel TCP connections is investigated using simulation experiments. However, for optimizing the number of parallel TCP connections, simulation-based approaches are inappropriate; i.e., it is quite difficult or, in most cases, impossible to apply simulation results for a parameter optimization. For optimizing the number of parallel TCP connections, some insight in the effect of parallel TCP connections on their performance is necessary. In [7, 9], simple analytic models of parallel TCP connections have been presented. However, those analytic models are not applicable for optimizing the number of parallel TCP connections since they do not capture the trade-offs in parallel data transfer, as we will explain in Section II.

In this paper, by particularly focusing on the number of parallel TCP connections and TCP socket buffer size, we quantitatively investigate their optimal parameter configurations and clarify performance limitations of GridFTP. We first derive a continuous-time model for GridFTP by aggregating multiple TCP continuous-time models [10]. Since TCP is a *feedback control* that changes the window size depending on packet loss probability in a network, we model multiple TCP connections as independent continuous-time SISO (Single-Input and Single-Output) systems. By combining these continuous-time TCP models, we then obtain a continuous-time model of GridFTP. Performance metrics in steady state (e.g., GridFTP goodput and packet loss probability) are derived using our

GridFTP model. By focusing on the number of TCP connections and TCP socket buffer size, we also derive the optimal parameter configuration for GridFTP and quantitatively show performance limitations of GridFTP.

Note that GridFTP supports two types of using parallel TCP connections: *parallel data transfer* and *striped data transfer* (see Section II). The parallel data transfer (from a single server to a client) is a special case of the striped data transfer (from multiple servers to a client). For brevity, we limit the scope of this paper to the parallel data transfer. However, our analytic approach can be directly applied to the case of the striped data transfer by simply differentiating round-trip times, R, in Section IV. Using a simple model enables us to derive several GridFTP performance metrics in a closed-form, which gives us more insights than with an unnecessarily complicated model.

The structure of this paper is as follows. First, Section II briefly explains GridFTP's major features (i.e., auto-negotiation of TCP socket buffer size and parallel data transfer), and discusses unresolved problems of GridFTP. Section III explains the definition of terms used in this paper and our approach for modeling GridFTP. Section IV analyzes steady state performance of GridFTP by aggregating multiple TCP continuous-time models, and derives the optimal parameter configuration of GridFTP. Section V compares analytic results with simulation ones for validating our approximate analysis. Finally, Section VI summarizes this paper and discusses future topics.

II. GridFTP

Standardization of GridFTP [2] as a common bulk data transfer protocol for the Grid has been currently proceeding in the Global Grid Forum (GGF) [11]. GridFTP is a protocol that extends FTP (File Transfer Protocol) [12-14], which was standardized in the IETF and has been widely used in the Internet. In addition to features of the original FTP, the following features are added to GridFTP: auto-negotiation of TCP socket buffer size, parallel data transfer, third-party control of data transfer, partial file transfer, security, and support for reliable data transfer.

In what follows, an overview of two major features of GridFTP, auto-negotiation of TCP socket buffer size and parallel data transfer, and unresolved problems of GridFTP is presented.

A. Auto-Negotiation of TCP Socket Buffer Size

In GridFTP, the server's TCP socket buffer size can be explicitly configured by the client with the SBUF (Set Buffer Size) command. In addition, TCP socket buffer size can be configured by negotiating between the GridFTP server and client using the ABUF (Auto-Negotiate Buffer Size) command. Almost all existing TCP implementations allocate a fixed-size (e.g., 64 [Kbyte]) TCP socket buffer, so throughput improvement can be expected when the TCP socket buffer size is appropriately configured according to the bandwidth-delay product of the network.

However, it has not been adequately studied how the ABUF command should be implemented. For instance, the ABUF command has not been implemented in the GridFTP implementation included in a Globus Toolkit [15]. Examples of the ABUF command implementation is discussed in [2, 5], which measure a round-trip time and available bandwidth of a network by generating measurement traffic between the GridFTP server and client.

However, more investigation on the ABUF command implementation is necessary. In a real network, the available bandwidth of a network varies with time, and a large amount of measurement traffic must be generated for accurately measuring the available bandwidth. Because of these reasons, active measurement approaches as discussed in [2, 5], which configure TCP socket buffer size based simply on the measured bandwidth-delay product, are inadequate for practical purposes.

B. Parallel Data Transfer

GridFTP can establish multiple TCP connections in parallel by using OPTS RETR or OPTS STOR command. Hence, a single file can be transferred via multiple TCP connections from/to a single or multiple GridFTP servers. Higher throughput than with a single TCP connection can be expected through aggregation of multiple TCP connections [6-8].

This can be explained by the following three reasons. First, larger bandwidth can be gained by aggregating multiple TCP connections when competing with other TCP connections in the TCP congestion avoidance phase [16]. This is because AIMD window flow control is adopted in the TCP congestion avoidance phase and data transfer can be performed better by aggregating multiple TCP connections in a network with a low packet loss probability. Second, the total TCP socket buffer size that can be used in a file transfer becomes large by aggregating multiple TCP connections. This is because the total of TCP socket buffer sizes is N times larger by aggregating N TCP connections. Third, the start-up of the transfer rate is accelerated in the TCP slow start phase by aggregating multiple TCP connections. In the slow start phase, the congestion window doubles for every round-trip time. Accordingly, the start-up for the transfer rate is N times faster by aggregating N TCP connections.

However, if the number N of aggregate TCP connections is too large, it results in decreased throughput for the following reasons. First, the window size per TCP connection decreases, and TCP timeout are more likely to occur. Second, the overhead required for the GridFTP server and client to process the TCP protocol stack increases. Accordingly, the optimal value for the number of aggregate TCP connections, N, must be determined according to several network conditions. However, it has not been fully investigated and is still an unresolved problem how to determine the number of parallel TCP connections N in various network environments.

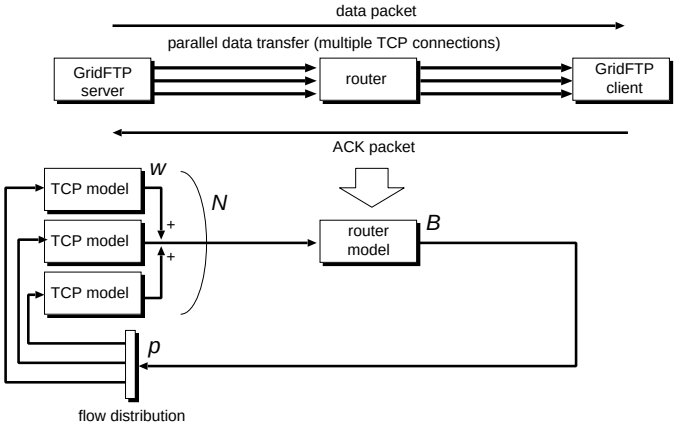

Fig. 1. Modeling GridFTP with parallel data transfer by aggregating TCP continuous-time models

III. Analytic Model

In this paper, modeling of GridFTP is performed using a network modeling technique proposed in [10, 17]. In what follows, a primary feature of GridFTP, parallel data transfer is modeled.

In what follows, a source host of data transfer is called a *GridFTP server* and a destination host of the data transfer is called a *GridFTP client*. For brevity, our analysis is limited to cases of GridFTP server-to-client file transfers. However, other cases of GridFTP client-to-server file transfers can be easily modeled using the same modeling approach.

GridFTP supports parallel data transfer, partial file transfer, and third-party control of data transfer, so there exists one or more GridFTP servers for a single GridFTP client. In our analysis, traffic on the control channel is assumed to be negligible, and only traffic on data channel is modeled.

Modeling GridFTP is performed as follows (Fig. 1). First, the GridFTP server is modeled by aggregating multiple continuous-time models of the TCP congestion control mechanism. When GridFTP performs parallel data transfer, multiple TCP connections are established between the GridFTP server and client. Accordingly, multiple TCP congestion control mechanism models are aggregated at the GridFTP server.

IV. Steady State Analysis

In this section, the optimal number of parallel TCP connections is derived by performing steady state analysis for our GridFTP model.

First, a case when the TCP socket buffer size W is larger than the bandwidth-delay product per TCP connection (i.e., TCP throughput $\times$ round-trip time) is considered. In this case, according to [10, 17], changes in the TCP window size $w(t)$ at time t including a TCP timeout mechanism are modeled using a fluid-flow approximation as

$$\begin{aligned}\dot{w} &= (1-p(t))\frac{w(t-R)}{w(t)R} \\ &\quad -p(t)\frac{2}{3}w(t)\frac{w(t-R)}{R}\{1-p_{TO}(t)\} \\ &\quad -p(t)\left\{\frac{4}{3}w(t)-1\right\}\frac{w(t-R)}{R}p_{TO}(t) \end{aligned} \quad (1)$$

where $p(t)$ is the packet loss probability in a network at time t, R is the round-trip time of TCP connections, and $p_{TO}(t)$ is the probability of detecting packet loss at time t due to TCP timeouts.

The packet loss probability and TCP window size in steady state are denoted by p^* and w^*, respectively. In steady state, $\dot{w} = 0$, $p(t) = p^*$, and $w(t) = w(t-R) = w^*$, so the following relationship is obtained from Eq. (1).

$$\frac{3 - p^*\ \left(3 + 2\,{w^*}^2\right) + p^*\ \left(3 - 2\,w^*\right)\ w^*\ p^*_{TO}}{3\,R} = 0 \quad (2)$$

In the above equation, TCP window size w^* in steady state is assumed to be larger than or equal to 3. Under this assumption, the probability of detecting packet loss due to TCP timeouts, p^*_{TO}, is given by [18]

$$p^*_{TO} \simeq \frac{3}{w^*} \quad (3)$$

By solving in Eqs. (2) and (3) for w^*, TCP window size w^* in steady state is obtained as

$$w^* \simeq \frac{1}{2}\left(-3 + \frac{\sqrt{6+21\,p^*}}{\sqrt{p^*}}\right) \quad (4)$$

TCP throughput T^* in steady state is then given by

$$T^* \equiv \frac{w^*}{R} \simeq \frac{1}{2\,R}\left(-3 + \frac{\sqrt{6+21\,p^*}}{\sqrt{p^*}}\right) \quad (5)$$

Since TCP socket buffer size is assumed to be larger than the bandwidth-delay product per TCP connection, packet loss is expected to occur only because of network congestion. Let N be the number of TCP connections, and B the bottleneck link bandwidth. Under these conditions, total throughput for all TCP connections $N\,T^*$ equals the bottleneck link bandwidth B in steady state. Therefore,

$$B = N\,T^* \simeq \frac{N}{2\,R}\left(-3 + \frac{\sqrt{6+21\,p^*}}{\sqrt{p^*}}\right) \quad (6)$$

is obtained from Eq. (5). By solving this equation for p^*, the packet loss probability in steady state is derived as

$$p^* \simeq \left(-2 + \frac{2\,B\,R}{N} + \frac{2}{3}\left(\frac{B\,R}{N}\right)^2\right)^{-1} \quad (7)$$

From the above equation, one can find that the packet loss probability in a network p^* in steady state is determined only by the bandwidth-delay product per TCP connection (i.e., $B\,R/N$).

When the packet loss probability in steady state is p^* and the throughput for all TCP connections is T^*, only a fraction p^* of

packets are discarded in the network. So, effective throughput (i.e., goodput) for a TCP connection G^* is given by

$$G^* \equiv T^*(1-p^*) \tag{8}$$

Next, a case when the TCP socket buffer size W is smaller than the bandwidth-delay product per TCP connection (TCP throughput $\times$ round-trip time). In this case, TCP window size cannot be fully increased regardless of the available bandwidth. Accordingly, TCP throughput T^* is limited by

$$T^* = \frac{W}{R} \tag{9}$$

When TCP socket buffer size W is smaller than the bandwidth-delay product per TCP connection, the bottleneck link bandwidth cannot be utilized at 100%. Provided that the packet loss caused by background traffic is negligible, the packet loss probability in steady state p^* is given by

$$p^* = 0 \tag{10}$$

Thus, the effective throughput for a TCP connection G^* in steady state becomes

$$G^* \equiv T^*(1-p^*) = T^* = \frac{W}{R} \tag{11}$$

From Eqs. (8) and (11), the GridFTP goodput (total effective throughput for all TCP connections) $\overline{G^*}$ is given as a function of the TCP socket buffer size W, the number of parallel TCP connections N, round-trip time R, and the bottleneck link bandwidth B; i.e.,

$$\overline{G^*} \simeq \min\left(\frac{N\,W}{R}, \frac{N\,(1-p^*)}{2\,R}\left(-3+\frac{\sqrt{6+21\,p^*}}{\sqrt{p^*}}\right)\right) \tag{12}$$

Thus, the optimal number of parallel TCP connections is obtained by determining N that maximizes Eq. (12); i.e., from Eq. (12), the optimal value of N is derived as

$$N = \left(3\,B\,R-3\,W-\sqrt{3}\,\sqrt{9\,B^2\,R^2-16\,B\,R\,W+7\,W^2}\right) \times\frac{B\,R}{9\,B\,R-6\,W} \tag{13}$$

On the contrary, Eq. (12) indicates that for a given number of parallel TCP connections, N, the TCP socket buffer size W should be large enough to maximize $\overline{G^*}$ in Eq. (12).

V. Numerical Examples

In what follows, the effect of the number of parallel TCP connections and TCP socket buffer size on GridFTP performance has been quantitatively investigated through several numerical examples of the steady state analysis.

First, GridFTP goodput (total goodput of all TCP connections) $\overline{G^*}$ in steady state is shown in Fig. 2. In this figure, we use the following parameters: the bottleneck link bandwidth $B = 8.3$ [packet/ms] (corresponding to 100 [Mbit/s] when a packet size is 1500 [byte]) and TCP socket buffer size $W = 64$ [Kbyte]. The round-trip time for TCP connections R and the number of parallel TCP connections N are varied. In addition, the packet loss probability for GridFTP in steady state is plotted in Fig. 3.

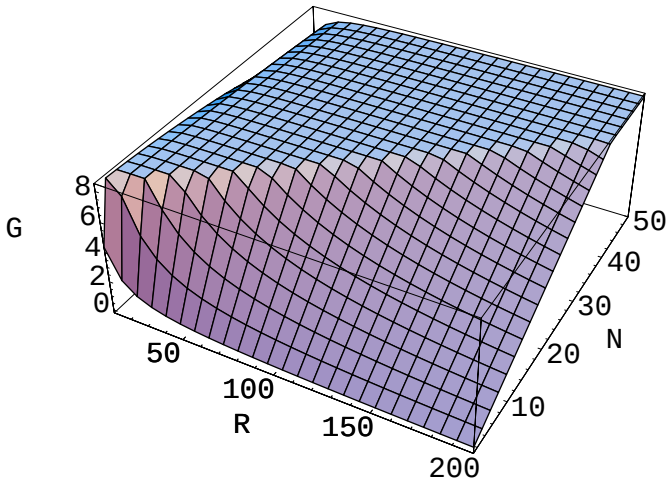

Fig. 2. GridFTP goodput $\overline{G^*}$ (effect of round-trip time R and the number of parallel TCP connections N) ($B = 8.3$ [packet/ms], $W = 64$ [Kbyte])

The followings regarding GridFTP goodput can be observed from these figures. First, from Fig. 2, the number of parallel TCP connections N should be large for utilizing the bottleneck link bandwidth at almost 100%. The required number of parallel TCP connections N for full link utilizing is almost proportional to the round-trip time for TCP connections. This phenomenon is in agreement with simulation results (see, e.g., [8]). Second, GridFTP goodput decreases slightly with the further increase in the number of parallel TCP connections. This tendency appears obviously for a small round-trip time. This is because the packet loss probability increases as the number of parallel TCP connections N increases and/or round-trip time for TCP connections R decreases [8], as can be seen from Fig. 3.

We then focus on the effect of the number of parallel TCP connections N and TCP socket buffer size W on GridFTP goodput and the packet loss probability in steady state. The GridFTP goodput and packet loss probability for different numbers of parallel TCP connections N and TCP socket buffer sizes W is shown in Figs. 4 and 5, respectively. The bottleneck link bandwidth is set to $B = 8.3$ [packet/ms], and the round-trip time for TCP connections at $R = 100$ [ms].

From Fig. 4, one can find that the TCP socket buffer size W and the number of parallel TCP connections N must be appropriately configured for fully utilizing the bottleneck link bandwidth. For instance, when the TCP socket buffer size W is small (e.g., 16 [Kbyte]), the number of parallel TCP connections N should be very large. Accordingly, configuring the TCP socket buffer size W to a sufficiently large value is desired for avoiding an extremely large number of parallel TCP connections N.

As the number of parallel TCP connections N increases, the packet loss probability for GridFTP increases according to Fig. 5. Also, the packet loss probability for GridFTP is

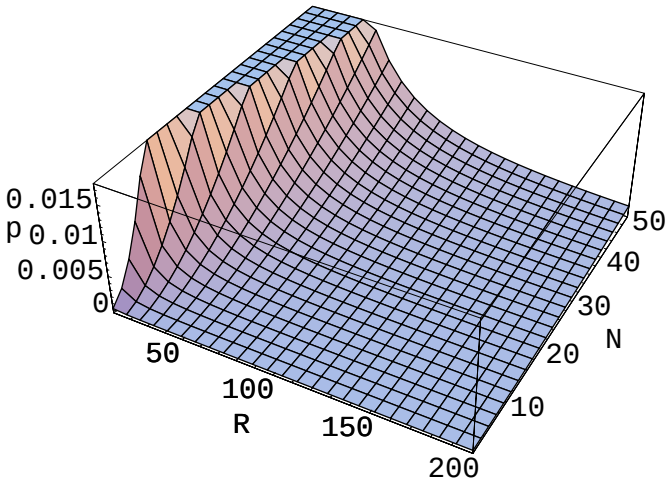

Fig. 3. GridFTP packet loss probability p^* (effect of round-trip time R and the number of parallel TCP connections N) ($B = 8.3$ [packet/ms], $W = 64$ [Kbyte])

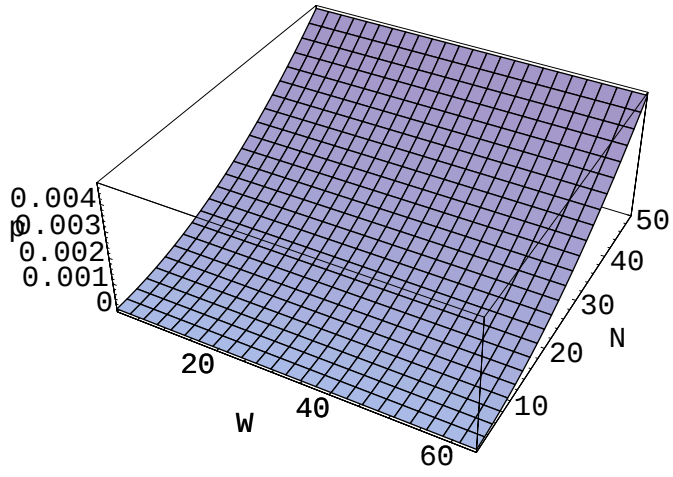

Fig. 5. GridFTP packet loss probability p^* (effect of the number of parallel TCP connections N and TCP socket buffer size W) ($B = 8.3$ [packet/ms], $R = 100$ [ms])

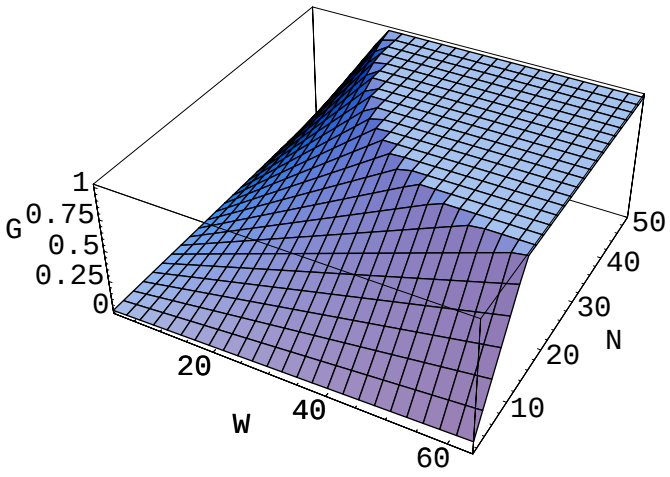

Fig. 4. GridFTP goodput $\overline{G^*}$ (effect of the number of parallel TCP connections N and TCP socket buffer size W) ($B = 8.3$ [packet/ms], $R = 100$ [ms])

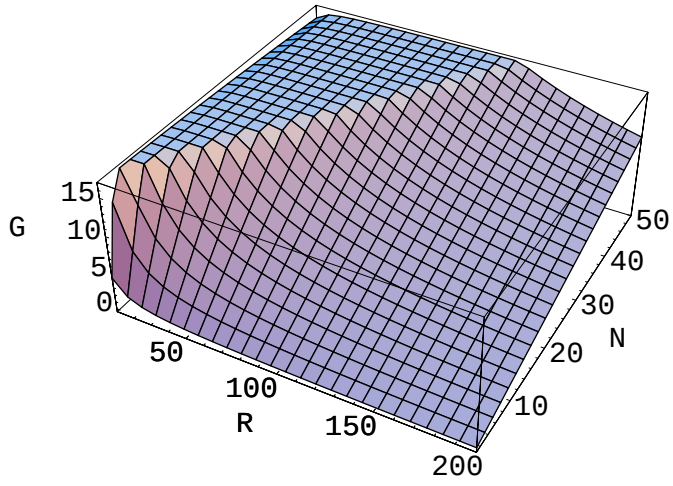

Fig. 6. GridFTP goodput ($N \times G^*$) (effect of round-trip time R and the number of parallel TCP connections N) ($B = 16.6$ [packet/ms], $W = 64$ [Kbyte])

independent of the TCP socket buffer size W. Based on these observations, a guideline for tuning GridFTP control parameter is to allocate as large a TCP socket buffer size W as possible, and to establish the number of parallel TCP connections N that can fully utilize the bottleneck link bandwidth. Note that the bandwidth-delay product of the network is in practice the upper-limit of the TCP socket buffer size W.

Next, effect of the bottleneck link bandwidth on the optimal parameter configuration of GridFTP is investigated. Figure 6 is a result with a larger bottleneck link bandwidth B ($B = 16.6$ [packet/ms]) than that of Fig. 2. From this figure, it can be found that the number of parallel TCP connections should be increased accordingly when the bottleneck link bandwidth is increased. However, by comparing Figs. 2 and 6, one can find that goodput degradation for GridFTP for a large number of parallel TCP connections is smaller for a larger bottleneck link bandwidth. This is because the bandwidth-delay product of each TCP connection increases as the bottleneck link bandwidth increases, and, consequently, TCP timeouts less likely to occur. This means that parameter configuration for GridFTP is simpler in a faster or wider-area network.

Finally, the validity of our approximation analysis is exam-

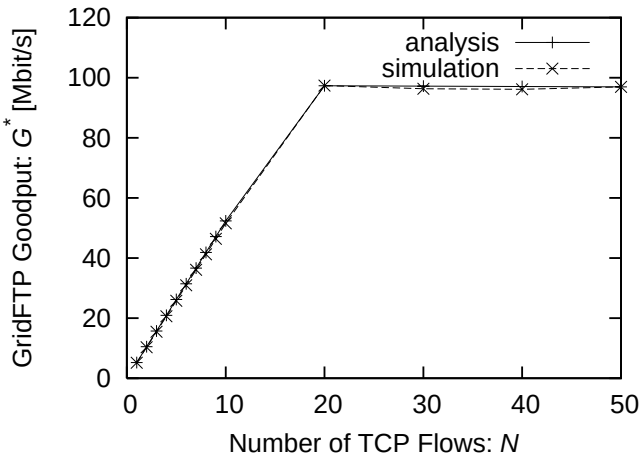

Fig. 7. Number of parallel TCP connections vs. GridFTP goodput ($B = 8.3$ [packet/ms], $\tau = 100$ [ms], $W = 64$ [Kbyte])

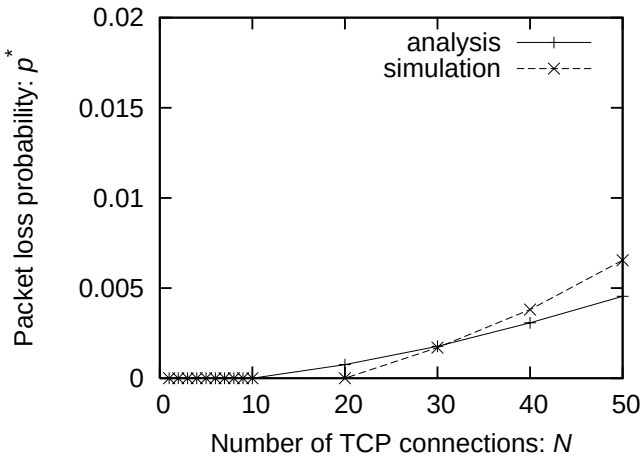

Fig. 8. Number of parallel TCP connections vs. GridFTP packet loss probability ($B = 8.3$ [packet/ms], $\tau = 100$ [ms], $W = 64$ [Kbyte])

ined through comparison of analysis and simulation results. ns-2 simulator (version 2.28) [19] is used for all simulations. A simple network topology of one hop is used in the simulation. The bottleneck link bandwidth is set to $B = 8.3$ [packet/ms], and the two-way propagation delay at $\tau = 100$ [ms] or $\tau = 20$ [ms]. The packet size is fixed at 1,500 [byte], and GridFTP server is modeled by aggregating FTP traffic sources. Every simulation is run for 60 [s] while changing the number of parallel TCP connections (i.e., the number of active FTP traffic sources), and the goodput and packet loss probability are measured.

First, simulation results for the two-way propagation delay $\tau = 100$ [ms] of the bottleneck link are shown in Figs 7 and 8. These figures show the goodput and packet loss probability for GridFTP when the number of parallel TCP connections is changed. Also, analytic results, which are calculated based on the average round-trip time for TCP connections obtained by simulation, are plotted. From these figures, one can find that the GridFTP goodput and packet loss probability are accurately estimated by our steady state analysis.

Simulation results for a small two-way propagation delay of the bottleneck link ($\tau = 20$ [ms]) are shown in Figs. 9 and 10. Compared to the case with larger two-way propagation delay (Figs. 7 and 8), it can be found that analytic results deviate from simulation results, in particular, when there are a large number of parallel TCP connections (e.g., $N = 50$). This phenomenon can be explained by the following reason. In a network with a small bandwidth-delay product and a large number of parallel TCP connections, window size for each TCP connection becomes small so that TCP timeouts are more likely to occur. However, the probability of detecting packet loss due to TCP timeouts, p^*_{TO}, is approximated by Eq. (3), which should not be used when the packet loss probability is large [18]. More accurate modeling of the probability of detecting packet loss due to TCP timeouts, p^*_{TO}, is necessary, but it is beyond the scope of the current paper.

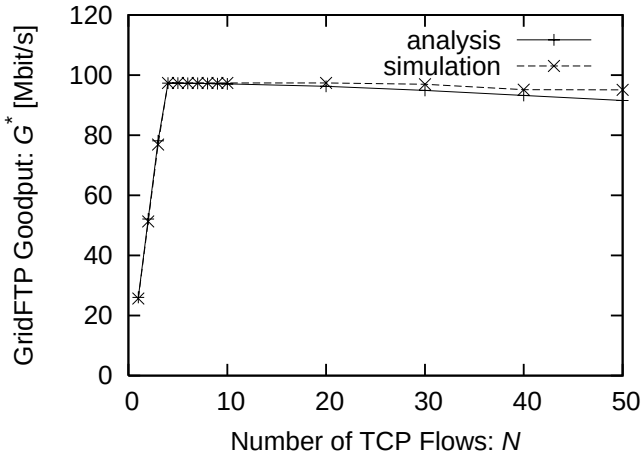

Fig. 9. Number of parallel TCP connections vs. GridFTP goodput ($B = 8.3$ [packet/ms], $\tau = 20$ [ms], $W = 64$ [Kbyte])

However, it should be noted that our steady state analysis is sufficiently usable in practice for optimizing control parameter for GridFTP. Namely, recall that the guideline for GridFTP parameter tuning is to use a sufficiently large TCP socket buffer size W and to configure the number of parallel TCP connections for fully utilizing the bottleneck link bandwidth. Hence, effect of the modeling error in an extremely larger number of parallel TCP connections can be negligible for parameter configuration purposes.

VI. Conclusions and Future Topics

In this paper, we have investigated the optimal parameter configuration for GridFTP, i.e., the number of parallel TCP connections and TCP socket buffer size, by performing a steady state analysis for GridFTP. First, a continuous-time model for GridFTP has been derived by aggregating multiple TCP continuous-time models. Steady state performance metrics (e.g., GridFTP goodput and packet loss probability)

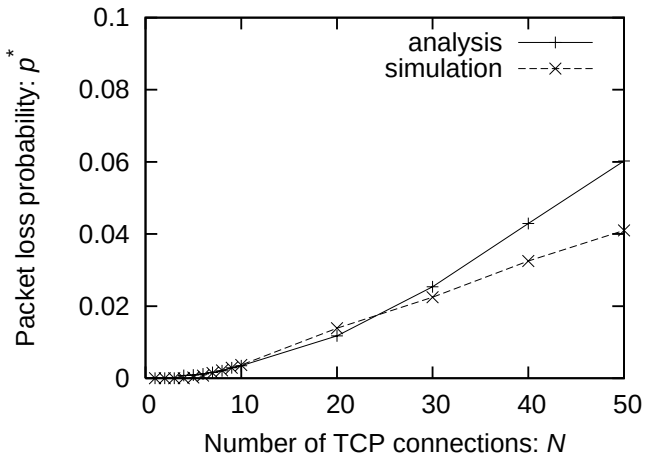

Fig. 10. Number of parallel TCP connections vs. GridFTP packet loss probability ($B = 8.3$ [packet/ms], $\tau = 20$ [ms], $W = 64$ [Kbyte])

have been derived using our GridFTP model. By focusing on the number of TCP connections and TCP socket buffer size, we have also derived the optimal parameter configuration for GridFTP, and quantitatively show performance limitations of GridFTP through analytic and simulation results. Our GridFTP parameter tuning guideline is to allocate as large a TCP socket buffer size W as possible (but no more than the bandwidth-delay product of the network) and to establish the number of parallel TCP connections N that can full utilize the bottleneck link bandwidth according to Eq. (13). We have also validated our approximate analysis by comparing simulation results with analytic ones.

Future research topics include improving the accuracy of our approximate analysis (e.g., accurate modeling of TCP timeout mechanism) and performing analysis of GridFTP in more generic network configurations with, for instance, background traffic and several different versions of TCP connections. Also important is applying our analytic results for automatically optimizing GridFTP performance. In [20], we are currently working on designing an automatic parameter configuration mechanism for GridFTP, which utilizes our analytic results and has compatibility with existing GridFTP servers.

REFERENCES

[1] J. Postel, "Transmission control protocol," *Request for Comments (RFC) 793*, Sept. 1981.

[2] W. Allcock *et al.*, "GridFTP: Protocol extensions to FTP for the Grid," *GGF Document Series GFD.20*, Apr. 2003, also available as http://www.gridforum.org/GFD.20.pdf.

[3] J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP buffer tuning," in *Proceedings of ACM SIGCOMM '98*, vol. 28, Oct. 1998.

[4] T. Dunigan, M. Mathis, and B. Tierney, "A TCP tuning daemon," in *Proceedings of SuperComputing: High-Performance Networking and Computing*, Nov. 2002.

[5] S. Thulasidasan, W. Feng, and M. K. Gardner, "Optimizing GridFTP through dynamic right-sizing," in *Proceedings of IEEE International Symposium on High Performance Distributed Computing*, June 2003.

[6] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, Nov. 2000.

[7] T. J. Hacker and B. D. Athey, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS)*, Aug. 2001.

[8] L. Qiu, Y. Zhang, and S. Keshav, "On individual and aggregate TCP performance," in *Proceedings of Internetl Conference on Network Protocols*, Oct. 1999, pp. 203–212.

[9] D. Lu, Y. Quao, P. Dinda, and F. Bustamante, "Modeling and taming parallel TCP on the wide area network," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, Apr. 2005.

[10] H. Ohsaki, J. Ujiie, and M. Imase, "On scalable modeling of TCP congestion control mechanism for large-scale IP networks," in *Proceedings of IEEE SAINT 2005*, Feb. 2005, pp. 361–369.

[11] "Global Grid Forum," http://www.ggf.org/.

[12] J. Postel and J. Reynolds, "File transfer protocol (FTP)," *Request for Comments (RFC) 959*, Oct. 1985.

[13] R. Elz and P. Hethmon, "FTP security extensions," *Request for Comments (RFC) 2228*, Oct. 1997.

[14] P. Hethmon and R. Elz, "Feature negotiation mechanism for the file transfer protocol," *Request for Comments (RFC) 2389*, Aug. 1998.

[15] "Globus Toolkit," available at http://www.globus.org/.

[16] S. Floyd, "Highspeed TCP for large congestion windows," *Internet Draft* `draft-ietf-tsvwg-highspeed-01.txt`, Aug. 2003.

[17] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 28–43, Feb. 2002.

[18] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of ACM SIGCOMM '98*, Sept. 1998, pp. 303–314.

[19] "The network simulator – ns2," available at http://www.isi.edu/nsnam/ns/.

[20] T. Ito, H. Ohsaki, and M. Imase, "Automatic parameter configuration mechanism for data transfer protocol GridFTP," submitted to *the 2006 International Symposium on Applications and the Internet (SAINT 2006)*, July 2005.