On Automatic Parameter Configuration Mechanism for Data Transfer Protocol GridFTP

Takeshi Ito, Hiroyuki Ohsaki and Makoto Imase

Graduate School of Information Science and Technology, Osaka University
1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan
E-mail: {t-itou; oosaki; imase}@ist.osaka-u.ac.jp

Abstract: In this paper, an automatic parameter configuration mechanism for GridFTP is proposed. This mechanism optimizes the number of parallel TCP connections by utilizing the analytic result of GridFTP throughput. The proposed mechanism first measures the network status (e.g., the goodput and the round-trip time of GridFTP data channels) at the GridFTP client. Based on these measurement results, it adjusts the number of parallel TCP connections for maximizing the GridFTP goodput. Three operational modes, MI (Multiplicative Increase), MI+ (Multiplicative Increase Plus), and AIMD (Additive Increase and Multiplicative Decrease) are proposed in this paper, each of which takes a different strategy for adjusting the number of parallel TCP connections. Performance of the proposed automatic parameter configuration mechanism is evaluated through simulation experiments. This paper reveals that the proposed automatic parameter configuration mechanism significantly improves the performance of GridFTP.
Keywords: automatic parameter configuration mechanism, Grid computing, GridFTP, parallel TCP connections

1. Introduction

In recent years, exploitation of the network in larger capacity and in wider region has rapidly been in progress. In an experimental testbed network, the bottleneck link bandwidth reaches several Tbit/s and the transmission delay between end hosts sometimes reaches a hundred msec. On the other hand, Grid computing, which connects geographically-distributed multiple computing resources through the network, has been receiving a great deal of attention [10]. Grid computing is expected to efficiently use computational resources that have not been sufficiently utilized and to enable large-scale scientific and engineering computation. However, such large-scale scientific and engineering computation requires large file transfer among computers through the network. For this purpose, there has been increasing demand for a high-speed data transfer protocol that can effectively transfer large volume data.

TCP (Transmission Control Protocol) has been widely used in the Internet to carry data traffic [14]. There are various versions of TCP, and the most popular ones are TCP version Reno (TCP Reno) and its variants [13]. TCP Reno adjusts the packet transfer rate to the network in the congestion avoidance phase by implementing an AIMD (Additive Increase and Multiplicative Decrease) window flow control. TCP Reno normally updates its window size every round-trip time, but the window size is halved when a packet loss is detected in the network. Thus, TCP Reno drastically decreases its throughput due to a small number of packet losses when the bandwidth delay product of the network is large. Consequently, it has been pointed out that TCP Reno has problems such as a difficulty in maintaining higher throughput in a network with high packet loss probability.

GridFTP has been proposed as a protocol to effectively transfer large volume data in Grid computing [4] [12]. GridFTP is designed to solve the existing TCP problems and has various additional features for this purpose. These features include, for instance, parallel data transfer using multiple TCP connections and automatic negotiation of TCP socket buffer size. It is known that the effectiveness of GridFTP depends largely on control parameter configuration such as the number of parallel TCP connections [8] [16]. However, examination has not been conducted sufficiently so far regarding how to configure GridFTP control parameters. For example, although a command is defined in the GridFTP protocol to specify the number of parallel TCP connections between GridFTP server and client, how to determine the number is not specified at all in the GridFTP protocol.

There have been several studies concerning the configuration method of the GridFTP control parameters [17] [11]. The literature [17] proposed the method to determine the required TCP socket buffer size by measuring the bandwidth delay product between GridFTP server and client. This is realized by extending the `SBUF` command in the GridFTP extension block mode. In [11], the optimal number of parallel TCP connections and TCP socket buffer size are derived by using a TCP fluid-flow model. However, to optimize the number of parallel TCP connections in GridFTP based on the analysis result in [11], the required information such as round-trip time of the TCP connections, the packet loss probability, and the available bandwidth of the network should be known in advance. For this reason, the analysis result in [11] cannot be directly applied to GridFTP in practice.

Therefore, this paper proposes a technique to optimize the number of parallel TCP connections in GridFTP (i.e., automatic parameter configuration mechanism) through the use of the analysis result in [11]. The proposed technique first measures the network status (e.g., the goodput and the round-trip time of GridFTP) at the GridFTP client. Then, it adjusts the number of parallel TCP connections based on the above information. Three operational modes (i.e., MI, MI+, and AIMD) are considered in this paper, of which each has a different al-

gorithm to adjust the number of parallel TCP connections. Performance of the proposed automatic parameter configuration mechanism is then assessed by simulation experiments. As a result, it is demonstrated that the proposed automatic parameter configuration mechanism substantially improves the performance of GridFTP.

The structure of this paper is as follows. Section 2 provides a general description of GridFTP as well as the explanation on parallel data transfer that is one of the notable features of GridFTP. Section 3 describes the design strategy and basic idea of the automatic parameter configuration mechanism, and then explains its algorithm in detail. Section 4 demonstrates the effectiveness of the automatic parameter configuration mechanism through simulation experiments. Lastly, Section 5 summarizes this paper and mentions future tasks.

2. GridFTP

GridFTP is a data transfer protocol, which is designed to effectively transfer large volume data in Grid computing [4] [12]. GridFTP is an extension to FTP (File Transfer Protocol) [15] [6] [9] that has been widely used, and was standardized in OGF (The Open Grid Forum, formerly known as GGF (The Global Grid Forum)) [3]. GridFTP, which uses TCP as its transport layer communication protocol, is designed to solve several problems of TCP. For example, besides the features of the existing FTP, it has additional features such as automatic negotiation of TCP socket buffer size, parallel data transfer, third-party control of file transfer, partial file transfer, security, and reliable data transfer [4] [12]. Most of these specific features of GridFTP are realized by a new transfer mode called extended block mode [4]. Currently, GridFTP server and client software conforming to GridFTP version 1 (GridFTP v1) [4] is implemented in the Globus Toolkit [1], which is the de facto standard middleware for Grid computing. However, in this specific GridFTP implementation, the feature of the automatic negotiation of TCP socket buffer size is not implemented and therefore a user must manually specify the number of parallel TCP connections for parallel data transfer. In addition, OGF has been discussing the problems with GridFTP v1 and has finished standardization of GridFTP v2 (version 2) as a solution to these problems [12]. Additional features have been incorporated in GridFTP v2. These features relax several limitations of the extended block mode in GridFTP v1, such that data transfer is restricted to a single direction and unable to open/close data channels in the midst of the data transfer. However, how to configure the GridFTP control parameters regarding parallel TCP connections has not been addressed even in the standardization of GridFTP v2.

2.1 Parallel Data Transfer

Multiple TCP connections can be established in parallel in GridFTP using `OPTS RETR` command (Fig. 1). With this mechanism, a single file can be transferred from a single server through multiple TCP connections. Higher throughput can be expected by aggregating multiple TCP connections in comparison to using a single TCP connection [16].

This can be explained by the following reasons: (1) by aggregating multiple TCP connections, larger bandwidth can be gained in the TCP congestion avoidance phase than those gained by other competing TCP connections [7]. This situation results from the fact that, in the TCP congestion avoidance phase, the AIMD window flow control is being executed, and therefore aggregated multiple TCP connections can transfer data more advantageously through the network with smaller packet loss probability. (2) by aggregating multiple TCP connections, the total TCP socket buffer size available to the file transfer increases. Aggregation of N TCP connections can utilize as N times TCP socket buffer size as that of a single TCP connection. (3) by aggregating multiple TCP connections, ramp-up time of the transfer rate in the TCP slow-start phase is shortened. In the slow-start phase, the congestion window doubles every round-trip time. For this reason, through the aggregation of N TCP connections the speed of the transfer rate increase becomes N times as fast as that of a TCP connection.

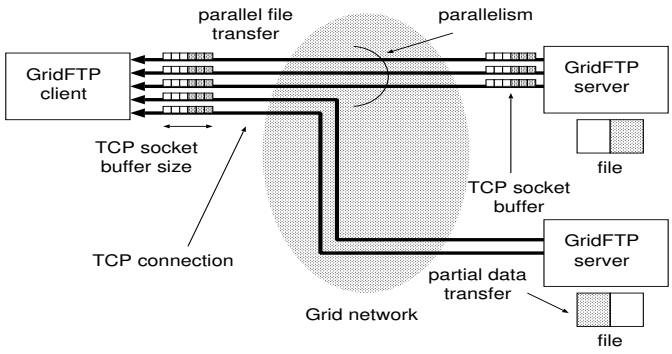

Fig. 1. Parallel data transfer in GridFTP

However, the throughput drops if the number of aggregate TCP connections, N, becomes too large, since this may cause the following situations: (1) the window size per TCP connection becomes smaller, so TCP timeout would frequently occur. (2) the overhead required for the server to process a TCP protocol stack would increase. Therefore, the optimal number of TCP connections, N, should be determined based on the network status. Nevertheless, how to optimize the number of parallel TCP connections, N, in GridFTP has not sufficiently been studied and still remains as an open issue.

3. Automatic Parameter Configuration Mechanism for GridFTP

3.1 Design Strategy

First, the basic principles for designing automatic parameter configuration mechanism for GridFTP is explained.

It is extremely important to provide the compatibility with existing GridFTP in designing an automatic parameter configuration mechanism for GridFTP. GridFTP is implemented in the Globus Toolkit and has been spreading rapidly in recent years. Since a number of GridFTP servers have already been in operation, it is preferable to realize the automatic parameter configuration mechanism at the side of GridFTP client. Additionally, it is also favorable to establish the automatic parameter configuration mechanism without changing the existing GridFTP protocol so as to interconnect with existing GridFTP servers. If the automatic parameter configuration is executed on the side of GridFTP client, it will be difficult to realize the automatic parameter configuration in the third-party transfer. Nonetheless, it should not be so problematic because most

of the transfers are executed between GridFTP servers and clients.

Next, it is desirable that the automatic parameter configuration mechanism for GridFTP can easily be installed in Grid computing environment. Generally, Grid computing is featured by the heterogeneity of computers and networks constituting Grid. Therefore, the automatic parameter configuration mechanism needs to operate in various computer environments as well as various network environments. For this reason, it is preferable for the automatic parameter configuration mechanism to be realized in the Grid middleware layer. In other words, it is important for the automatic parameter configuration mechanism to avoid using any functionality specific to certain operating systems or network devices on the computers. Currently, most of the computers use the transport layer communication protocol based on TCP version Reno. Therefore, it is reasonable to assume the transport layer communication protocol to be TCP version Reno.

3.2 Basic Idea

In [11], the GridFTP goodput in steady state is approximately derived as

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

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

where G is the GridFTP goodput, N is the number of parallel TCP connections, W is the TCP socket buffer size for each TCP connection, B is the bandwidth for the bottleneck link, R is the round-trip time of the TCP connections, and p^* is the packet loss probability in the network. Equation (1) indicates that the GridFTP goodput G is a convex function of the number N of parallel TCP connections. Thus, the number of parallel TCP connections, N, should be selected so as to maximize the GridFTP goodput G. However, Eq. (1) also indicates that other parameters such as the bottleneck link bandwidth B and the TCP connection round-trip time R should be known in order to determine the optimal number of N. This paper proposes a mechanism to automatically configure the number of parallel TCP connections on the side of GridFTP client by using the analysis result as formulated in Eq. (1).

The basic idea is to search for the number of parallel TCP connections, N, which maximizes Eq. (1) while measuring the network status (i.e., the GridFTP goodput and the round-trip time) on the side of GridFTP client. Equation (1) shows that the GridFTP goodput is upper-bound by $N\,W/R$ when the number of parallel TCP connections, N, is too small. Therefore, the idea is to firstly fix the number of parallel TCP connections, N, and transfer a chunk of the file to be transferred and then measure the goodput G as well as the round-trip time R on the side of GridFTP client. Whether the current number of parallel TCP connections, N, is larger than the optimal value can be assessed by evaluating the relationship between G and $N\,W/R$ after transferring the chunk. Based on this result, the number of parallel TCP connections, N, is updated, and another chunk is then transferred. By repeating this procedure, the number of parallel TCP connections can automatically be configured so as to maximize the GridFTP goodput.

3.3 Measuring Network Status

This section explains how to measure the network status on the side of GridFTP client. Note that the GridFTP goodput G, the TCP socket buffer size W, and the round-trip time R, are measurable in the middleware layer.

The GridFTP goodput can be calculated from the size of a chunk and its transfer time, which is measured by transferring the chunk in the extended block mode [4]. Specifically, a chunk is transferred by `ERET` or `ESTO` command in GridFTP's extended block mode while measuring its response time, T. Since the response to `ERET` or `ESTO` command is returned when the chunk transfer is completed [4], the GridFTP goodput can be calculated as $G = X/T$ from the size of the transferred chunk, X, and the response time, T.

The TCP socket buffer size on the side of GridFTP client can easily be obtained with the use of the socket API. GridFTP has a feature to statically configure the socket buffer size (i.e., `SBUF` command) [4], and the TCP socket buffer size on the side of the server can be obtained using this feature (more precisely, it can be configured to the value specified by the GridFTP client). Either the TCP socket buffer size of GridFTP client or that of GridFTP server, which has the smaller value, will be used as W in Eq. (1).

The round-trip time R can be obtained by measuring the command response time in the GridFTP control channel. Specifically, the response time R_i is measured when commands such as `USER`, `PASS`, `SITE`, `FEAT`, `TYPE`, `MODE`, `SIZE`, `OPTS`, `NOOP`, and `PBSZ` are executed by GridFTP client. It can be assumed that GridFTP server would instantly complete the process of these commands. For this reason, the round-trip time R can be approximated by the response time to these commands. More precisely, the running average of response times; i.e., $R \leftarrow R(1-\rho) + R_i\rho$ is used as the estimated round-trip time R where $\rho(0 < \rho < 1)$ is a parameter. As explained above, measurement is possible by a passive measurement method that does not burden the network by using just control packets or data packets of GridFTP.

On the other hand, among the variables that are included in Eq. (1), measurement of the bottleneck link bandwidth B and the packet loss probability p^* in the network is difficult on the side of GridFTP client. At present, it is difficult to measure the bottleneck link bandwidth B or the packet loss probability p^* at the middleware layer using a passive measurement method. Should features specific to certain operating systems or network devices on the computers be used, it would become possible to measure the bottleneck link bandwidth B or the packet loss probability p^*. However, ease of installation and/or deployment in the Grid computing environment should be compromised. Likewise, even if the bottleneck link bandwidth B or the packet loss probability p^* could also be measured by employing an active measurement method (e.g., generating UDP/ICMP packet for measurement in the middleware layer), the compatibility with existing GridFTP servers would be impaired.

3.4 Adjusting the Number of Parallel TCP Connections

This section explains how to adjust the number of parallel TCP connections, N, so as to maximize the GridFTP goodput after measuring the network status. The automatic parameter configuration mechanism proposed in this paper transfers chunks of a file while measuring the GridFTP goodput, the TCP socket buffer size W, and the round-trip time R. Based on this information, the number of parallel TCP connections, N, will be adjusted. For adjusting the number of parallel TCP connections, N, three operation modes as described below are discussed.

• MI (Multiplicative Increase) Mode

The MI mode starts from a small number of parallel TCP connections, N, and multiplicatively increases it until the TCP socket buffer size becomes not the bottleneck that impedes the GridFTP goodput. In the following, let N_0 denote the initial value of the number of parallel TCP connections, $\gamma(> 1)$ denote a multiplicative increase factor, and $\eta(0 < \eta \leq 1)$ denote a control parameter. The operation algorithm of the MI mode is as follows.

(i). Initialize the number of parallel TCP connections.

$$N \leftarrow N_0 \tag{3}$$

(ii). Transfer a chunk and then measure the GridFTP goodput G and the round-trip time R.

(iii). If the following inequality is satisfied, conjecture the TCP socket buffer size is the bottleneck and proceed to the step (iv). Otherwise, terminate the algorithm.

$$G > \eta \times \frac{N\,W}{R} \tag{4}$$

(iv). Increase the number of parallel TCP connections as follows, and return to the step (ii).

$$N \leftarrow \gamma \times N \tag{5}$$

• MI+ (Multiplicative Increase Plus) Mode

The MI+ mode starts from a small number of parallel TCP connections, N, and multiplicatively increases it until the TCP socket buffer size becomes not the bottleneck that impedes the GridFTP goodput. It then optimizes the number of parallel TCP connections, N, using the result of the steady state analysis (Eq. (1)). In the following, let N_0 denote the initial value of the number of parallel TCP connections, $\gamma(> 1)$ denote a multiplicative increase factor, and $\eta(0 < \eta \leq 1)$ denote a control parameter. The operation algorithm of the MI+ mode is as follows.

(i). Initialize the number of parallel TCP connections.

$$N \leftarrow N_0 \tag{6}$$

(ii). Transfer the chunk, and then measure the GridFTP goodput G and the round-trip time R.

(iii). If the following inequality is satisfied, conjecture the TCP socket buffer size is the bottleneck and proceed to the step (iv). Otherwise, proceed to the step (v).

$$G > \eta \times \frac{N\,W}{R} \tag{7}$$

(iv). Increase the number of parallel TCP connections as follows, and return to the step (ii).

$$N \leftarrow \gamma \times N \tag{8}$$

(v). Numerically derive the optimal value of the number of parallel TCP connections, which maximizes the second term of the right-hand side of Eq. (1). Configure the number of parallel TCP connections to that value and terminate the algorithm.

• AIMD (Additive Increase and Multiplicative Decrease) Mode

The AIMD mode starts from a small number of parallel TCP connections, N, and additively increases it if the TCP socket buffer size is the bottleneck or otherwise multiplicatively decrease it. In the following, let N_0 denote the initial value of the number of parallel TCP connections, $\alpha(> 1)$ denote an additive increase factor, $\beta(0 < \beta < 1)$ denote a multiplicative decrease factor, and $\eta(0 < \eta \leq 1)$ denote a control parameter. The operation algorithm of the AIMD mode is as follows.

(i). Initialize the number of parallel TCP connections.

$$N \leftarrow N_0 \tag{9}$$

(ii). Transfer the chunk and then measure the GridFTP goodput G and the round-trip time R.

(iii). If the following inequality is satisfied, conjecture the TCP socket buffer size is the bottleneck and proceed to the step (iv). Otherwise, proceed to the step (v).

$$G > \eta \times \frac{N\,W}{R} \tag{10}$$

(iv). Increase the number of parallel TCP connections as follows, and return to the step (ii).

$$N \leftarrow N + \alpha \tag{11}$$

(v). Decrease the number of parallel TCP connections as follows, and return to the step (ii).

$$N \leftarrow N(1 - \beta) \tag{12}$$

3.5 Adjusting the Chunk Size

For measuring the GridFTP goodput accurately and quickly, it is important to appropriately determine the chunk size in each chunk transfer.

For accelerating the adjustment of the number of parallel TCP connections, it is desirable to keep the chunk size as small as possible. However, if the chunk size is too small, the goodput of each TCP connection cannot be measured accurately because of TCP's characteristics [5].

For solving this problem, the automatic parameter configuration mechanism predicts the GridFTP goodput of the next chunk transfer, and dynamically configures the chunk size so that the chunk transfer time becomes as fixed as possible.

In what follows, $G(N)$ denotes the GridFTP goodput measured at the chunk transfer with N TCP connections, and N_{-k} the number of parallel TCP connections used for the k-th previous chunk transfer.

The automatic parameter configuration mechanism determines the chunk size X as follows. The automatic parameter configuration mechanism predicts the GridFTP goodput of the next chunk transfer to be $G(N_{-1})G(N_{-1})/G(N_{-2})$ from the ratio of past chunk transfers, and determine the chunk size as

$$X \leftarrow G(N_{-1})\frac{G(N_{-1})}{G(N_{-2})}\Delta, \tag{13}$$

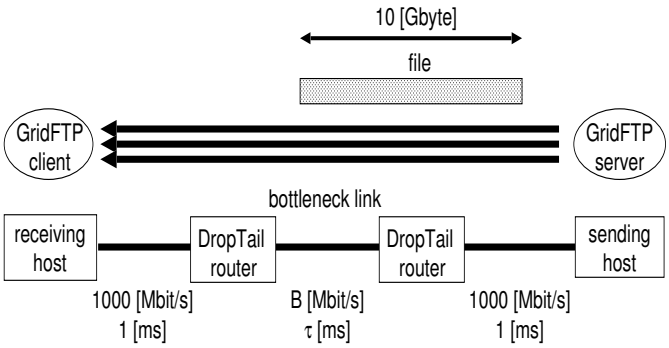

Fig. 2. Network topology used in simulation

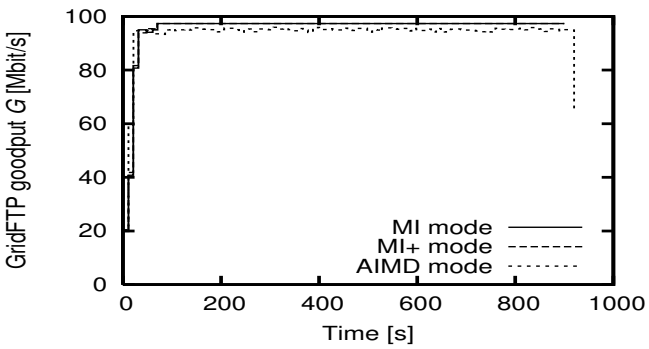

Fig. 3. Evolution of GridFTP goodput G ($B = 100$ [Mbit/s], $\tau = 10$ [ms], $W = 64$ [Kbyte])

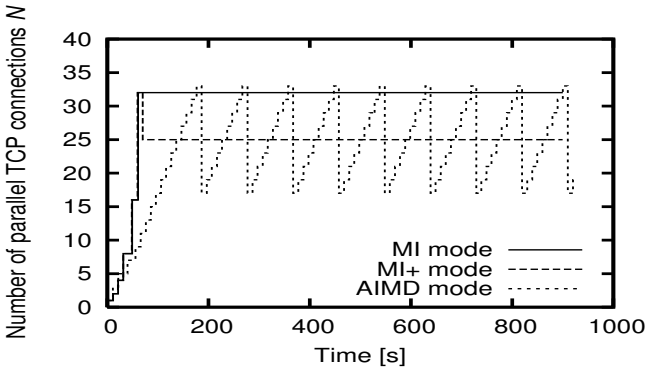

Fig. 4. Number of parallel TCP connections N ($B = 100$ [Mbit/s], $\tau = 10$ [ms], $W = 64$ [Kbyte])

where Δ is a control parameter, which is the target value of the chunk transfer time.

Note that, at the time of the first chunk transfer, the GridFTP goodput $G(N_{-1})$ and $G(N_{-2})$ are unknown. Hence, the chunk size X is determined as follows.

$$X \leftarrow \frac{N_0 W}{R}\Delta \tag{14}$$

At the time of the second chunk transfer, the GridFTP goodput $G(N_{-2})$ is unknown. So the chunk size X is determined as follows.

$$X \leftarrow N/N_{-1}\, G(N_{-1})\Delta \tag{15}$$

4. Simulation

In this section, the effectiveness of the proposed automatic parameter configuration mechanism is evaluated by simulation experiments. Figure 2 shows the network topology used in simulation. In this network topology, a GridFTP server and a GridFTP client are connected via two DropTail routers. A file of 10 [Gbyte] is transferred from the GridFTP server to the client. Using the proposed three operation modes of the automatic parameter configuration mechanism, the number of parallel TCP connections is dynamically changed.

The ns-2 simulator (version 2.28) [2] was used for the simulation. The following parameters were used in all simulations unless otherwise stated: the bottleneck link bandwidth B is 100 [Mbit/s]; the propagation delay τ is 10 [ms]; the buffer size of DropTail router is 1,000 [packet]; the TCP socket buffer size W is 64 [Kbyte]; the TCP packet length is 1,500 [byte]; the parameter ρ is 0.1; the initial value of the number of parallel TCP connections N_0 is 1; the target value of the chunk transfer time Δ is 10 [s]; the control parameter η is 0.8; the multiplicative increase factor γ of the MI mode and MI+ mode is 2.0; the additive increase factor α of the AIMD mode is 2.0; and multiplicative decrease factor β of the AIMD mode is 0.5.

Figures 3 and 4 show the evolution of the GridFTP goodput G and the number of parallel TCP connections, N, adjusted by the automatic parameter configuration mechanism, respectively. These figures plot the GridFTP goodput and the adjusted number of parallel TCP connections, which were measured when every chunk transfer was completed.

It can be seen from Figs. 3 and 4 that both the MI mode and MI+ mode perform well. For both the MI mode and MI+ mode, the number of parallel TCP connections exponentially increases after the start of the file transfer, and converges to 32 and 25, respectively, after approximately 70 [s] from the start of the file transfer. The GridFTP goodput at this point is approximately 97 [Mbit/s] for the MI mode and MI+ mode.

These results show that GridFTP can sufficiently use the available bandwidth of the network by using the MI mode or MI+ mode. These also show that the rise of the number of parallel TCP connections is slower in the AIMD mode compared to both the MI mode and MI+ mode, in which the number of parallel TCP connections, N, increases linearly. However, since the transient performance of the AIMD mode largely depends on the additive increase factor α and multiplicative decrease factor β, further examination is necessary to study the effect of these control parameters.

Next, the GridFTP goodput for a different bottleneck link bandwidth B is shown in Fig. 5. In this figure, the TCP socket buffer size W is set to 1 [Mbyte]. For comparison purposes, the GridFTP goodput with a single TCP connection is also included in the figure, showing poor performance when the bottleneck link bandwidth is large. This figure indicates that the bottleneck link bandwidth can be almost fully utilized by using the automatic parameter configuration mechanism. However, utilization of the bottleneck link is slightly degraded as the bottleneck link bandwidth becomes large. This is because TCP connections take time to fully utilize the bottleneck link bandwidth when the link bandwidth is large. Consequently, the automatic parameter configuration mechanism takes time to find the optimal number of TCP connections, leading less GridFTP goodput.

The GridFTP goodput when changing the propagation delay τ is shown in Fig. 6. This figure shows that with the MI mode and MI+ mode, the GridFTP goodput hardly degrades even

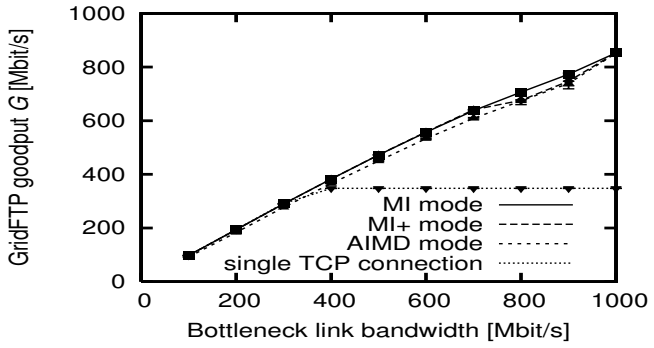

Fig. 5. Effect of the bottleneck link bandwidth B on GridFTP goodput G ($\tau = 10$ [ms], $W = 1$ [Mbyte])

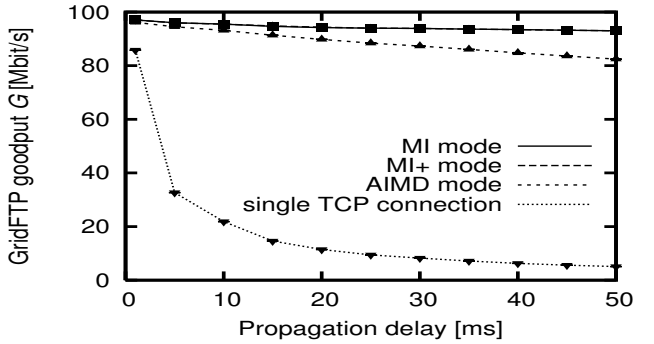

Fig. 6. Effect of propagation delay τ on GridFTP goodput G ($B = 100$ [Mbit/s], $W = 64$ [Kbyte])

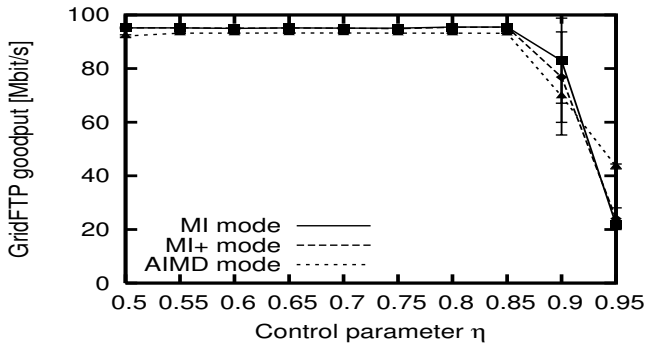

Fig. 7. Effect of control parameter η on GridFTP goodput G ($B = 100$ [Mbit/s], $\tau = 10$ [ms],$W = 64$ [Kbyte])

for a large propagation delay. On the contrary, with the AIMD mode, the GridFTP goodput decreases almost linearly as the propagation delay becomes large.

Finally, Fig. 7 shows the GridFTP goodput when changing the control parameter η of the automatic parameter configuration mechanism. This figure shows that in the MI mode, MI+ mode, and AIMD mode, the goodput of GridFTP is almost independent of the control parameter η as long as η is less than 0.85.

From these observations, it is concluded that the MI mode and MI+ mode show a good performance. Namely, with the MI mode or MI+ mode, GridFTP with the automatic parameter configuration mechanism can realize high goodput regardless of the bottleneck link bandwidth and the propagation delay. On the contrary, with the AIMD mode, GridFTP can achieve high goodput when the propagation delay is not large. However, the AIMD mode requires careful configuration of its control parameters α and β.

5. Summary and Future Works

In this paper, the automatic parameter configuration mechanism for GridFTP has been proposed, mainly focusing on the parallel data transfer feature for GridFTP. The proposed automatic parameter configuration mechanism transfers a file between GridFTP server and client as chunks, and then measures the GridFTP goodput and the round-trip time. Based on the measured GridFTP goodput and the round-trip time, the number of parallel TCP connections is adjusted so as to maximize the GridFTP goodput. As operational modes to adjust the number of parallel TCP connections, three modes MI, MI+, and AIMD have been proposed, and their effectiveness has been evaluated by simulation experiments. The simulation results demonstrate that the proposed automatic parameter configuration mechanism significantly improves the GridFTP goodput and thus enables to effectively utilize the available bandwidth of the network.

As a future work, simulation experiments under various network configurations to identify a parameter range that optimizes the performance of the proposed automatic parameter configuration mechanism should be conducted. In simulation experiments, performance of the automatic parameter configuration mechanism has been evaluated when the file size is sufficiently large. When the file size is small, it is expected that the performance of the automatic parameter configuration mechanism is significantly affected by configuration of the update interval of the number of parallel TCP connections and the chunk size. Hence, investigation on the optimal parameter configuration for those control parameters is necessary. Fairness among GridFTP sessions sharing the same bottleneck link should also be evaluated. In a general network with multiple GridFTP sessions, maintaining fairness among those GridFTP sessions is an important issue. Regarding fairness among GridFTP sessions, it is expected that the AIMD mode may perform better than the MI and MI+ modes, but more through investigation is definitely required.

Acknowledgments

We would like to deeply express our appreciation and gratefulness to both Mr. Masayuki Murata and Mr. Hideyuki Yamamoto of the Graduate School of Information Science and Technology, Osaka University, for joining meaningful discussions and theoretical arguments in our process of conducting this research.

This work is supported by the NAREGI (National Research Grid Initiative) Project from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] Globus Toolkit, available at `http://www.globus.org/`.

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

[3] Open Grid Forum, `http://www.ogf.org/`.

[4] W Allcock, et al., GridFTP: Protocol extensions to FTP for the Grid. OGF Document Series GFD.20, 2003, Also available as `http://www.ggf.org/documents/GFD.20.pdf`.

[5] N Ehsan and M Liu, Analysis of TCP transient behavior and its effect on file transfer latency. Proceedings of IEEE International Conference on Communications (ICC2003) 2003, Vol. 26, pp. 1806–1811.
[6] R Elz and P Hethmon, FTP security extensions. Request for Comments (RFC) 2228, 1997.
[7] S Floyd, HighSpeed TCP for large congestion windows. Request for Comments (RFC) 3649, 2003.
[8] T J Hacker, B D Athey and B Noble, The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS) 2002, pp. 434–443.
[9] P Hethmon and R Elz, Feature negotiation mechanism for the file transfer protocol. Request for Comments (RFC) 2389, 1998.
[10] I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kauffman, San Francisco, 1999.
[11] T Ito, H Ohsaki and M Imase, On parameter tuning of data transfer protocol GridFTP in wide-area Grid computing. Proceedings of Second International Workshop on Networks for Grid Applications (GridNets 2005) 2005, pp. 415–421.
[12] I Mandrichenko, W Allcock and T Perelmutov, GridFTP v2 protocol description. OGF Document Series GFD.47, 2005, Also available as `http://www.ggf.org/documents/GFD.47.pdf`.
[13] J Padhye and S Floyd, On inferring TCP behavior. ACM SIGCOMM Computer Communication Review, Vol. 31, No. 4, 2001, pp. 287–298.
[14] J Postel, Transmission control protocol. Request for Comments (RFC) 793, 1981.
[15] J Postel and J Reynolds, File transfer protocol (FTP). Request for Comments (RFC) 959, 1985.
[16] L Qiu, Y Zhang and S Keshav, On individual and aggregate TCP performance. Proceedings of Internetl Conference on Network Protocols 1999, pp. 203–212.
[17] S Thulasidasan, W Feng and M K Gardner, Optimizing GridFTP through dynamic right-sizing. Proceedings of IEEE International Symposium on High Performance Distributed Computing 2003, pp. 14–23.

Author Bios

Takeshi Ito received B.E. degree in the Information and Computer Sciences from Osaka University. He also received the Master of Information Science and Technology degree from Osaka University, Osaka, Japan, in 2006. He is currently a Ph.D. candidate at Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Japan. His research work is in the area of Grid networks.

Hiroyuki Ohsaki received the M. E. degree in the Information and Computer Sciences from Osaka University, Osaka, Japan, in 1995. He also received the Ph. D. degree from Osaka University, Osaka, Japan, in 1997. He is currently an associate professor at Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Japan. His research work is in the area of traffic management in high-speed networks. He is a member of IEEE and Institute of Electronics, Information, and Computer Engineers of Japan (IEICE).

Makoto Imase received his B.E. and M.E. degrees in information engineering from Osaka University in 1975 and 1977, respectively. He received D.E. degree from Osaka University in 1986. From 1977 to 2001, he was engaged Nippon Telegraph and Telephone Corporation (NTT). He has been a Professor of Graduate School of Information Science and Technology at Osaka University since 2002. His research interests are in the area of information networks, distributed systems and graph theory. He is a member of IPSJ, JSIAM, and IEICE.