DDMG : A Data Dissemination Mechanism for Grid Environments

Hyung Jinn Kim

University of Science and Technology, Republic of Korea

Jongsuk Ruth Lee

Korea Institute of Science and Technology Information, Republic of Korea

Abstract
Until these days, the data dissemination transfer was not a fundamental feature in Grid environments. However, as the use of data-intensive applications become quantitatively and scalably increasing among Grid communities, the requirement of an effective data dissemination transfer becomes prominent. In a restricted environment like LAN or special purpose network environment the multicast transfer technology was typically used for such data transfer case. However, the limited data transfer performance of the multicast transfer, and the obligation of a special hardware setting makes it difficult to implement in a common Grid environments. Therefore, in this paper we propose DDMG(Data Dissemination Mechanism for Grid), an effective data dissemination mechanism for Grid environments. DDMG consists of an optimized P2P(Peer-to-Peer) transfer mechanism and Globus XIO library to improve the performance in data dissemination and to support heterogeneous protocols of the Grid environments. We will also evaluate the performance of DDMG by comparing with a typical unicast transfer.

Keywords:
Grid, Data Dissemination, P2P(Peer-to-Peer), Globus XIO(Globus eXtended Input/Output)

1. Introduction

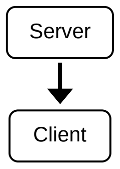
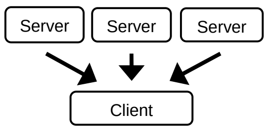
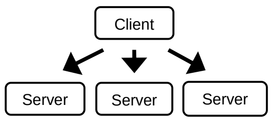
The major Grid applications like high energy physics, biology and meteorology are known as data intensive applications[1,2,3,4]. For example, high energy physics produces several petabytes of data and the meteorology produces several hundred of gigabytes of data. In such applications, transferring large scale of bulk data to local sites or other remote resources for processing is fundamental. So-called data Grid provides essential infrastructure for such application[5].

Data Grid is a technology which unifies the diverse and geologically dispersed storages by offering to users a common interface. Specially, data Grid aimed to offer a practical and efficient data related service in WAN(Wide Area Network) environment. To transfer a large scale of bulk data efficiently and reliably in a Grid environment, common protocols like HTTP(Hyper-Text Transfer Protocol) and FTP(File Transfer Protocol) are not appropriate. Therefore, Grid researchers have tried to investigate a new protocol suitable for Grid[6].

GridFTP(Grid File Transfer Protocol) is developed by ANL(Argone National Laboratory) for data transfer in Grid environments[7]. It is an extension of standard FTP protocol, and supports parallel data transfer, TCP buffer/window size auto balancing, striped mode transfer, and many other abilities to make data transfer more efficient in Grid environments. Beside of these data transfer ability, it also supports Kerberos[8] – a popular security mechanism – and GSI(Grid Security Infrastructure)[9] – the major security mechanism of a common Grid environment. Major Grid projects like EGEE(Enabling Grid for E-Science)[10] or Teragrid[11] project use GridFTP as a major data transfer protocol.

However, GridFTP does not cover all features needed for transferring a bulk data in Grid environments. Table 1 shows the major features of GridFTP. As shown in the table, GridFTP provides diverse features for the singlepoint-to-singlepoint and the multipoint-to-singlepoint data transfer, but does not provide any features for the singlepoint-to-multipoint transfer. The requirements for the singlepoint-to-multipoint transfer in Grid environments are rising. For example, high energy physics produces several petabytes of data each year and that data is needed to be replicated throughout the world[2]. And another example is TeraGyroid project[12]. It also needs to transfer throughout the world a bulk of data which generated by a large scale simulation machine. Up to now these projects have used GridFTP based on the high performance network for transferring data to the remote site. However, as the scale of the projects is getting larger, the requirement for an efficient data transfer mechanism is raising. Therefore, we have proposed a data transfer mechanism called DDMG(Data Dissemination Mechanism for Grid environment) which can enhance the efficiency and the performance of data transfer in Grid environments.

Table 1 : Features of GridFTP

Type of Data Transfer	Diagram of Data Transfer	Features of GridFTP
Singlepoint To Singlepoint		- Parallel data transfer - Automatic negotiation of TCP buffer/window sizes - Partial file transfer - Third-party control of data transfer
Multipoint To Singlepoint		- Striped data transfer
Singlepoint To Multipoint		- No existing feature

The rest of this paper is organized as fellows. In section 2, we will give an overview of some related researches done for efficient data dissemination, and an overview of a framework library for supporting multi protocol which is widely used in Grid environments. In section 3, we introduce the design and the implementation of DDMG. In section 4, we evaluate the performance of DDMB in LAN and WAN environments. Finally, we close this paper with conclusion and an outlook of future works in section 5.

2. Related Works

In this section, we introduce some multicasting technologies for Grid, and give an overview of the advantages and disadvantages of these technologies. We also introduce a P2P(Peer-to-Peer) data sharing mechanism which the usage have radically grown in the past few years. We finish this section by introducing a framework library named Globus XIO[13] which eases the support of multiple protocols.

2.1 Data Dissemination Mechanisms

2.1.1 Multicast Transfer Mechanism

Multicast transfer was usually used to disseminate data in LAN(Local Area Network) environment. It was attracted by the ability to decrease the network transfer load. Many researchers working on the multicast transfer have developed many multicast transfer protocols. However, most of internet router do not support the multicast transfer to avoid its abuse, and only a few protocols were developed for considering Grid environments.

TCP-XM(TCP eXtended to support Multicast)[14] is a multicast protocol developed for Grid environments. TCP-XM automatically detects the ability of multicast transfer on the underlying network, and it transparently uses multicast when available. TCP-XM is a modified TCP protocol which ensures the reliable data transfer and it's implementation was based on lwTCP(lightweight TCP/IP stack)[15] which can be implementable in diverse platforms without any kernel modifications. TCP-XM protocol like other multicast transfer protocol can decrease the traffic load of data transfer. However, the performance is not as good as a normal TCP transfer.

Other multicast protocols such as MDP(Multicast Dissemination Protocol)[16], NORM(Nack-Oriented Reliable Multicast)[17] and LGMP(Local Group-based Multicast Protocol)[18] were not developed in consideration of Grid environments. However, these protocols have a potential features usable in Grid. MDP is a framework protocol to support a reliable multicasting. It employs NACKs(Negative Acknowledgements) for discovering the data loss in data transfer. NORM is developed to enable the efficient data transfer across the heterogeneous IP networks and protocols. NORM also uses the NACKs mechanism for the reliable data transfer and was standardized by the reliable multicast transport protocol working group of IETF. LGMP is a reliable multicast protocol based on the idea defined by the local group concept. LGMP forms dynamically one or more sub-groups of nodes, and specifies a group controller node in each sub-group to manage the loss data discovery and various feedback processings.

All of these protocols are developed with the availability of open source implementation and the support of several platforms. They also provide a solution for the reliability and scalability of the transfer. And the implementations of these protocols are sufficiently matured to use in the real world[19]. However, the performance of these protocols is not as good as a normal TCP transfer, and they cannot be implemented in WAN environment without any multicast support of the router.

2.1.2 P2P(Peer-to-Peer) as a Data Sharing Mechanism

Nowadays P2P technology is widely used for sharing files via the internet to eliminate bottlenecks and to decrease the download times[20,21]. A general unicast transfer that consists of a server/client concept has a disadvantage that all the transfer services are centralized to the server. Therefore, an IO bottleneck occurs at the server side when the number of client grows. It causes the decrement of the transfer performance. Many multicast transfer technologies are developed to solve this bottleneck problem. However, most of the multicast transfer implementation has the performance limitation compared with a normal unicast transfer and the multicast transfer has a difficulty in applying to the common internet environment, because most of internet routers do not support the multicast transfer.

The P2P data sharing technology extends the server/client concept by adding to the client the ability of a server function. Therefore, when the client receives a part of a complete data, it can also redistribute to other clients. Therefore, the network load can be scattered and in the client's view the transfer occurred by several other client can have the same effect of a parallel transfer. The P2P data sharing mechanism can provide all of these effects without any hardware settings.

2.2 Globus XIO (Globus eXtended Input/Output)

Globus XIO[13] is a framework protocol library which provides a common interface for several heterogeneous protocols. It provides a simple and extensible I/O API and was developed for the following two main purposes.

- Providing a common interface to all transfer protocols for the Grid environments
- Minimizing the development time for creating or designing an interface for a new protocol

Up to now, to develop an application which supports multiple transfer protocols, we must use several protocol APIs which generally are different each other. Developing with these several protocol APIs lead a messy code and a long development time. And after the deployment of this application, extending new protocol could be also a lot of work. However, Globus XIO provides a common interface for protocols which can minimize the complexity of the development with different protocol API and can provide a convenience when extending different protocols. The common interface provided by Globus XIO can also decrease the development time for creating or designing an interface of a new protocol and help to concentrate on the core protocol development.

Figure 1 shows the structure of Globus XIO. In this figure '*User API*' describe the user interface which has OCRW(open/close/read/write) function, and '*Framework*' describes the core library of Globus XIO. '*Driver Stack*' is the stack where different driver could be registered and the '*Driver*' represents the module which has the core functionality of each protocol. The '*Driver*' can be classified into '*transfer driver*' and '*transform driver*'. The '*transfer driver*' is a driver which is concerned about data transfer and the '*transform driver*' is a driver which is concerned about data processing. For example, TCP driver is a transfer driver which has the functionality of transferring data through TCP protocols, and GSI driver is a transform driver which has the functionality of encrypting data. In driver stack, just one transfer driver must exist and one or more transform driver may exist.

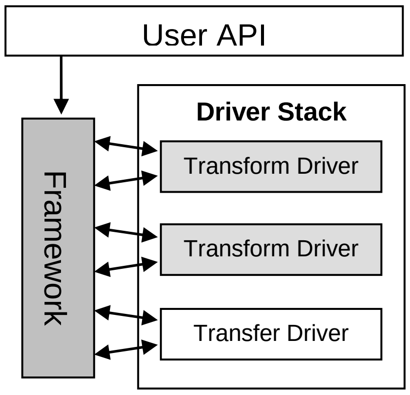


Figure 1. The Structure of Globus XIO

3. Design and Implementation of DDMG

The DDMG(Data Dissemination Mechanism for Grid) proposed in this paper aims to enhance the data dissemination performance in term of the transfer speed. For this purpose an optimized P2P data sharing mechanism is adopted, to induce the performance boost by an efficient use of network usage. The transfer of a bulk data in DDMG is managed by the client and transferred from a client to one or more servers uni-directionally. And all the data transfer is wrapped by Globus XIO to ease the support of multiple protocols.

3.1 Data Transfer Mechanism of DDMG

A general P2P data sharing mechanism is developed to share data among the undefined amounts of peers within the undefined period of times. Each peer has to schedule each data transfer which causes a transfer overlap among peers and decreases the data transfer efficiency. Surplus exchange of information (i.e. data stats, peer information, etc) are also inevitable and can potentially increase the network loads. Therefore, some optimizations are needed to the P2P data sharing mechanism for an efficient data transfer.

The DDMG aims to increase the data transfer efficiency by defining the receiving nodes before the data transfer occurs. Advanced defining the receiving nodes helps to optimize the data dissemination, dismiss unnecessary features like a peer discovery mechanism, and minimize the informational message exchange between nodes.

Figure 2 shows the data transfer mechanism of DDMG. The sender gets from the user the filename of data to be sent, the addresses of receivers and the protocol name to be used to transfer data to each receivers. Then it sends the filename and the number of receiver information to each receiver. From this received information, each receiver initializes sockets that will be used to receive data from other receivers and send the initialized sockets information to the sender. At the same time, the sender divides the data into the number of receivers. After receiving all the socket information from each receiver, the sender retransmits the information of the divided data and the information of initiated socket to each receiver. From that information, each receiver initiates the connection to each other and sends the ready message to the sender. After receiving the ready message from all the receivers, the sender sends each part of data to each receiver. And each receiver relay the transmitted data to other receivers. When the data transfer is completed, each receiver sends to the sender a '*transfer completed'* message. After receiving all '*transfer completed*' message from each receiver, the sender alarms to the user the transfer completion and finishes the transfer process (see appendix A for more detailed process).

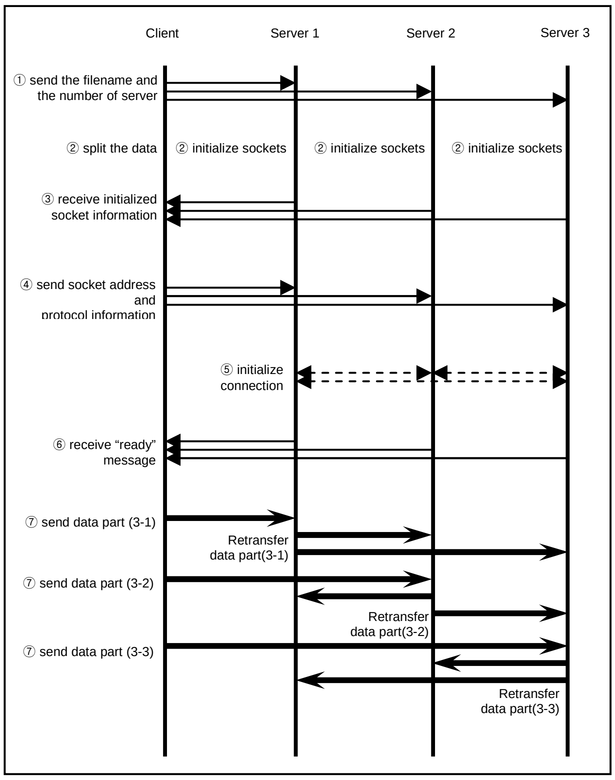

Client
Server 1
Server 2
Server 3
① send the filename and the number of server
② split the data
② initialize sockets
② initialize sockets
② initialize sockets
③ receive initialized socket information
④ send socket address and protocol information
⑤ initialize connection
⑥ receive "ready" message
⑦ send data part (3-1)
Retransfer data part(3-1)
⑦ send data part (3-2)
Retransfer data part(3-2)
⑦ send data part (3-3)
Retransfer data part(3-3)

Figure 2. Diagram of Data Flow in DDMG

3.2 Implementation of DDMG

The DDMG was implemented in a Linux environment with standard C language and used Globus XIO library from Globus Toolkit version 4.0.2. The Transfer architecture of this implementation consists of a control channel and a transfer channel. The control channel is used for exchanging information needed for the data transfer and the transfer channel is used to transfer the bulk data. These channels are wrapped with the Globus XIO framework, for helping to change protocol easily. While a new protocol is added in the system, an appropriate setting of this protocol can be dynamically added to the plug-in directory. Figure 3 shows the architecture of DDMG implementation.

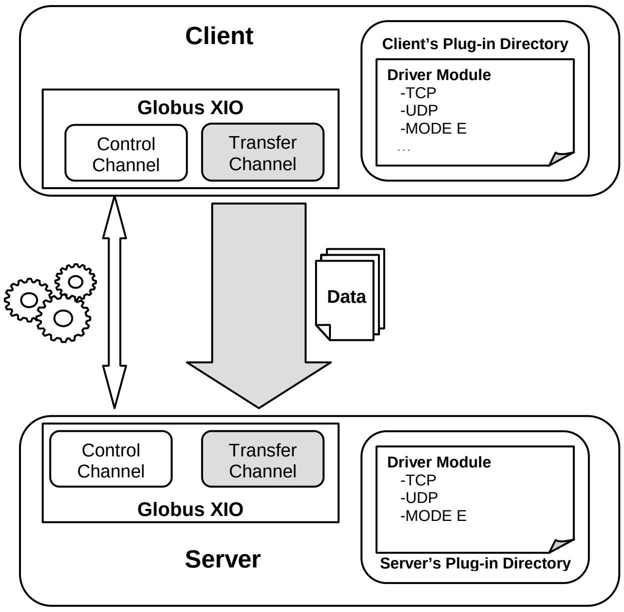


Figure 3. The Architecture of DDMG

4. Performance Evaluation of DDMG

We have evaluated the performance of DDMG in LAN and WAN environments. For the evaluation in LAN environment, we have used 1Gbps network backbone, and for WAN environment we have used four sites located at Seoul, Daejeon, Kwangju, and Pohang which are connected with KREONET(Korea Research Environment Open Network)[22] which is the Korean Scientific Network Backbone. The evaluation aims to compare the data transfer time of DDMG with a general unicast transfer in respect of the number of receivers.

In LAN environment, we have evaluated the performance in two cases. The first case is comparing the performance when the sender and the receiver nodes are in a high performance network (1Gbps) environment. And the second case is comparing the performance when the sender and the receiver nodes are in a medium performance network (100Mbps) environment which is more common in the real world. For the evaluation in WAN environments, we have also tested in two cases. The first is comparing the performance when the sender is directly connected to the network backbone. And the second case is to comparing when the sender is not directly connected to the network backbone.

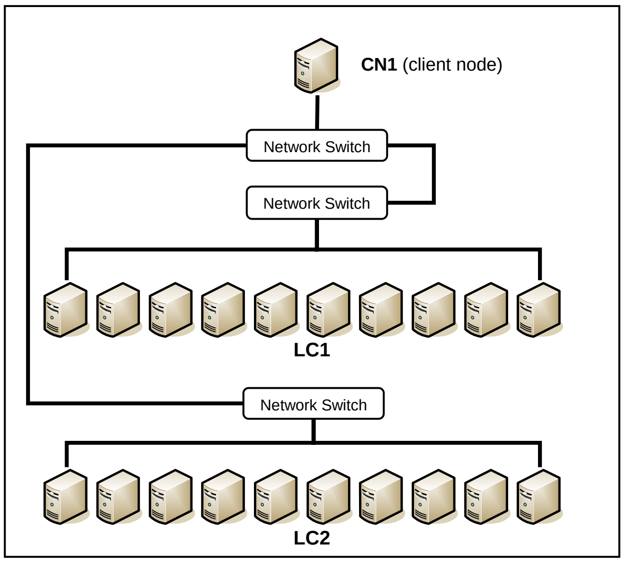
We have used TCP protocols for each evaluation. To compare with DDMG we have coded locally a unicast transfer implementation. The unicast implementation consists of simple send() and receive() functions of the Globus XIO library for eliminating the performance difference by implementing the different APIs. Each experimental result is obtained by averaging the result of each test which repeated 10 times.

4.1 Performance Evaluation in LAN Environment

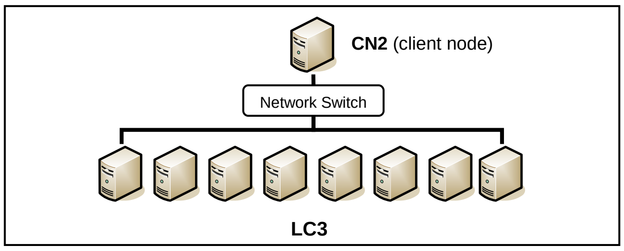
To evaluate the performance in LAN environment we organized a testbed like the Figure 4. The operation system of each node is RedHat Linux 9 and an E-IDE interface hard drive with 7200rpm rotational speed is used. For the high performance network evaluation (see ⓐ of Figure 4) we organized twenty receiver nodes and one sender node with 1Gbps NIC(Network Interface Card) installed each. And for the medium performance network evaluation (see ⓑ of Figure 4), we organized eight receiver nodes and one sender node with 100Mbps NIC installed each. The test purpose is to measure the transfer time for sending one giga bytes of bulk data in respect of the number of receiver nodes.

Figure 5 shows the difference of the transfer time between DDMG implementation and a general unicast transfer implementation in the high performance network. As shown in the figure, the transfer time of a general unicast transfer grows up proportionally in respect of the number of receiver nodes. However, the transfer time of DDMG keeps a steady result regardless of the number of receiver nodes. It also shows that the performance of a unicast transfer is better than DDMG when the number of receiver nodes is less than 5 nodes. The poor performance with DDMG is caused by the IO bottleneck when accessing the disk with DDMG mechanism (see appendix B).

Figure 6 shows the difference of the transfer time between DDMG implementation and a general unicast transfer implementation in medium performance network. Similarly to the previous evaluation, the transfer time of the unicast transfer grows up proportionally but the growth rate is steeper. However, the transfer time of DDMG grows up too but the growth rate is very low compared to the unicast transfer.



ⓐ High Performance(1Gbps) Network Testbed



ⓑ Medium Performance(100Mbps) Network Testbed

Figure 4. Testbed for the Evaluation in LAN Environment

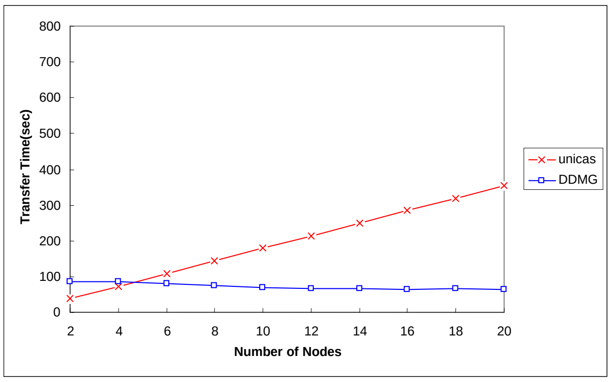


Figure 5. Transfer Time in the High Performance LAN Environment

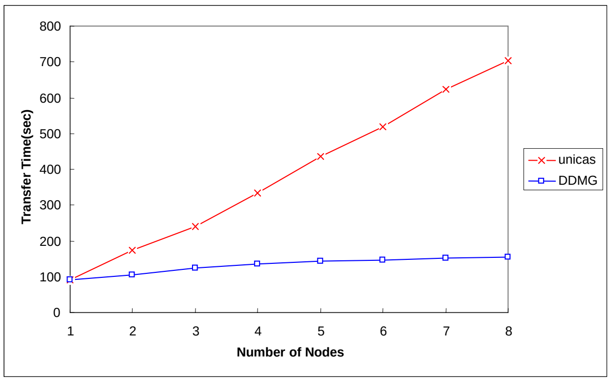


Figure 6. Transfer Time in the Medium Performance LAN Environment

4.2 Performance Evaluation in WAN Environment

We organized a testbed like the Figure 7 to evaluate the performance in WAN environment. The operation system of each node is RedHat Linux 9, and an E-IDE interface hard drive with 7200rpm rotational speed is used. Except the DJC2 node which is not connected directly in the WAN backbone, all other nodes have 1Gbps NIC installed. However, the DJC2 node has 100Mbps NIC installed. We used SLC, KJC, and PHC nodes for receivers, DJC1 for directly connected sender node, and DJC2 for indirectly connected sender node. We have evaluated by adding one node in each site respectively, and measured the transfer time for sending one giga bytes of bulk data.

Figure 8 shows the difference of the transfer time between DDMG and the unicast transfer when the sender node is connected to the WAN backbone directly. As shown in the figure, the transfer time of the unicast transfer and DDMG are very similar to those of the high performance network evaluation in LAN environment. The transfer time of unicast transfer grow up proportionally and those of DDMG keep a steady rate regardless of the number of receiver nodes.

Figure 9 shows the difference of the transfer time between DDMG, and the unicast transfer when the sender node is not connected to the WAN backbone directly. As shown in the figure, the transfer time of unicast transfer grows up with a steep growth rate. However, the transfer time of DDMG keeps a steady rate regardless of the number of receiver nodes.

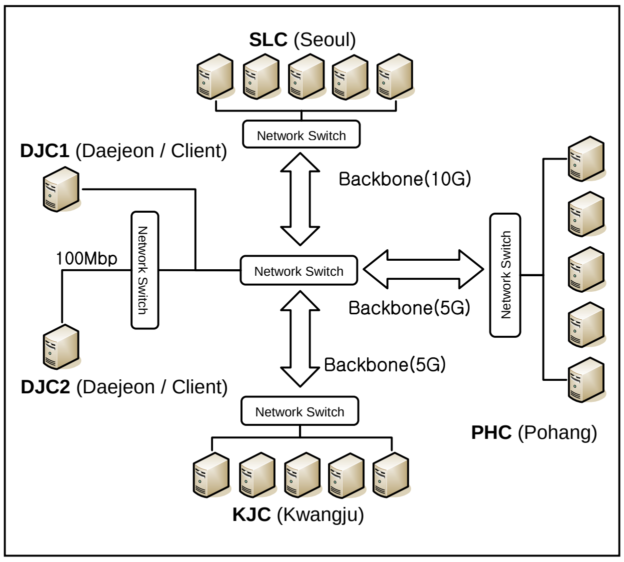


Figure 7. Testbed for the Evaluation in WAN Environment

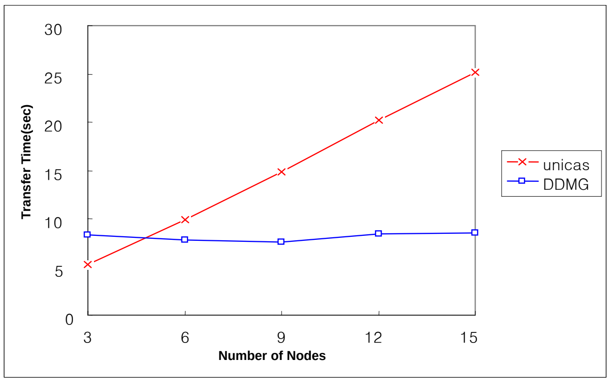


Figure 8. Transfer Time when the Sender is Connected to the Backbone Directly

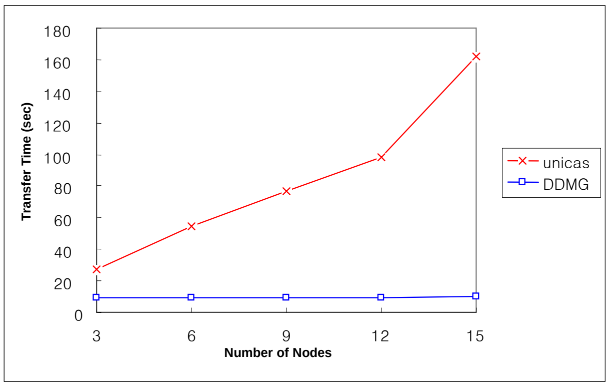


Figure 9. Transfer Time when the Sender is Not Connected to the Backbone Directly

To conclude of all these evaluation done in LAN and WAN environments, the transfer performance (in respect of transfer speed) of the unicast implementation is better then DDMG when network bandwidth is sufficiently large and the number of receivers are very small. However, when the network bandwidth is limited or large number of receivers participates in the data transfer DDMG shows a better transfer performance compared with unicast transfer, and the performance difference between DDMG and unicast transfer grows up proportionally in respect of the number of receiver nodes.

5. Conclusion and Future Works

Nowadays the requirement of data dissemination in Grid environments is growing. For the data dissemination, the multicasting technology is widely used in LAN or special purposed network environments. However, the transfer performance is not as good as a unicast transfer and without a special hardware setting it is difficult to apply it in WAN environment. Therefore, applying a multicast technology in Grid environment was very limited. In this paper, we have proposed a data dissemination mechanism named DDMG which used an optimized P2P data sharing mechanism to enhance the efficiency in data transfer and Globus XIO library for supporting multiple protocols.

We have evaluated the performance of DDMG implementation by comparing with a general unicast transfer implementation in LAN and WAN environments. The result shows that the transfer time of unicast transfer grows up proportionally in respect of the number of receivers. And it also showed that the performance of unicast transfer is better then DDMG when the network bandwidth is sufficiently large and the number of receivers are small. However, when the network bandwidth is limited or the number of receivers is not small, the DDMG implementation shows a better performance than those of unicast implementation and the performance difference among DDMG and unicast transfer grows up proportionally in respect of the number of receivers.

Even though DDMG shows a good performance, it needs some additional features for using in the real world. The DDMG splits the data at the same size according to the number of the receiving nodes. Therefore, the data transfer can be severely depended by a receiving node that has a radically small network bandwidth compared with other receiving nodes. To overcome this problem, a dynamic data transfer balance mechanism which manages automatically the data transfer rate of the receiving nodes is needed. Another feature needed for a functional use is a fault tolerant mechanism. This feature can prevent the transfer fault caused by a connection losts of a receiving node. After all these features are implemented, the DDMG can be widely used in many Grid applications which need data replications like the areas of high energy physics, meteorology researches and so on.

Acknowledgments

We would like to thanks to the members of Grid Computing Research Team of KISTI(Korea Institute of Science and Technology Information) who offered the accounts of all the testbed for testing the DDMG and for the detailed and insightful comments.
This work was supported by the research grant of supercomputing center, Korea Institute of Science and Technology Information.

References

[1] The Grid Physics Networks (GriPhyN) project, http://www.griphyn.org
[2] The Large Haron Collider (LHC) Project, http://lhc.web.cern.ch/lhc
[3] The Particle Physics Data Grid Project, http://www.cacr.caltech/edu/ppdg
[4] B. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data Management and Transfer in High-Performance Computational Grid Environments," Parallel Computing, Vol. 28, Issue 5, pp. 749-771, 2002.
[5] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Application, Vol. 23, no. 3, pp. 187-200, 2000.
[6] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of Supercomputer Applications, Vol. 15, no. 3, 2001.
[7] W. Allcock, J. Bseter, J. Bresnahan, S. Meder, P. Plaszczak, and S. Tuecke, "GridFTP: Protocol Extensions to FTP for the Grid," GFD-R.020, January 2004.
[8] B. C. Neuman, Theodore Ts'o, "Kerberos: An Authentication Service for Computer Networks," IEEE Communications, Vol, 32, Issue 9, pp. 33-38. September 1994.
[9] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, "Security for Grid Services," High Performance Distributed Computing Proceedings, pp. 48-57, June 2003.
[10] EGEE project, http://eu-egee.org
[11] TeraGrid project, http://teragrid.org
[12] S. Pickels et al, "The TeraGyroid Project," Proceedings of the 10th Globual Grid Forum(GGF10) : Workshop on Case Studies on Grid applications, March 2004.
[13] W. Allcock, J. Bresnahan, R. Kettimuthu, and J. Link, "The Globus eXtensible Input/Output System (XIO): A Protocol Independent IO System for the Grid," 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 4, pp. 179, 2005.
[14] K. Jeacle and J. Crowcroft, "Reliable High-speed Grid Data Delivery using IP Multicast," Proceedings of All Hands Meeting, 2003.
[15] A. Dunkels, "Minimal TCP/IP Implementaition with Proxy Support," Technical Report SICS-T-2001/20-SE, Swedish Institute of Computer Science, Feburary 2001.
[16] J. P. Macker, "The Multicast Dissemination Protocol (MDP) Toolkit," Proceedings of IEEE MILCOM, Vol. 1, pp. 626-630, 1999.
[17] B. Adamson, C. Bormann, M. Handley, and J. Macker, "NACK-Oriented Reliable Multicast Protocol (NORM)," RMT Working Group Internet Draft, January 2004.
[18] M. Hofmann, "Scalable Multicast Communication in Wide Area Networks," PhD Thesis, University of Karlsruhe, Germany, 1998.
[19] M. P. Barcellos, M. Nekovee, M. Daw, J. Brooke, and S. Olafsson, "Reliable Multicast for the Grid: A Comparison of Protocol Implementations," e-Science All-Hands Meeting, Nottingham, UK, September, 2004.
[20] D. Plonka. "Napster Traffic Measurement," http:://net.doit.wisc.edu/data/Napster, March 2000.
[21] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), pp. 315-328, December 2002.
[22] http://kreonet.re.kr

Hyung Jinn Kim received the BSc degrees in Mechanical Engineering from Hankuk Aviation University in 2004. From 2005, he attends the University of Science and Technology majoring Grid computing to acquire his MSc degree. His research interest are Grid computing especially in data management and data transfer, Operating System Architecture and Extreme Programming.

Jongsuk Ruth Lee is a senior researcher in Grid Technology Research Department, Supercomputing Center, Korea Institute of Science and Technology Information(KISTI), South Korea and also a assistant professor in Grid/Supercomputing Department, University of Science and Technology of Korea. She received a PhD in Computer Science from the University of Canterbury, New Zealand. Her research interests are Grid computing, Grid middleware, parallel/distributed computing, and parallel/distributed simulation

Appendix A

The pseudocode of the server and client implementation of DDMG

Table A-1. Pseudocode of the Client Implementation of DDMG

```
Get the information below from the user;
  - data filename;
  - addresses of the servers (receiving nodes);
  - protocol names of each server (receiving nodes);

For(each server) {
  Send the information below to the server;
      - data filename;
      - number of server;
  }

Divide the data into the number of server;

Calculate the memory address of each data part from the
divided data;

For(each server) {
  Receive the initialized socket address from the server;
  }

For(each server) {
Send the below information to server;
    - initialized socket information of each server;
    - protocol information of each server;
    - memory address of sending data part;
  }

For(each server) {
  Receive the "ready" message from each server;
  Start sending the appropriate data part to the server;
}
```

Table A-2. Pseudocode of the Server Implementation of DDMG

```
Receive the below information from the client;
  - data filename;
  - number of server;

For(number of server) {
  Initialize socket connections;
  }

Send to the client the initialized socket information;

Receive the information below from the client;
  - initialized socket information of each server;
  - protocol information of each server;
  - memory address of receiving data part;

File initialization for receiving process;

For(each other servers) {
  Connect to the server;
  Send the memory address of the receiving data part;
  }

For(each initialized socket) {
  Receive the memory address of the receiving data part
from other servers;
  Get ready to receive data part;
  }

Send the "ready" message to the client;

Receive data part from the client;

For(each of other servers) {
  Retransfer the received data from the client to other
servers;
  }
```

Appendix B

The evaluation of the performance degradation by the disk IO bottleneck in DDMG mechanism

This evaluation aims to explain the disk IO bottleneck occured in the DDMG transfer. We organized a testbed consisted of three nodes like Figure B-1. The operating system of each node is RedHat Linux 9, use an E-IDE hard drive with 7200rpm rotational speed (see Table B-1) and 1Gbps NIC is installed to each node. These three nodes are connected to each other with 1Gbps network switch.

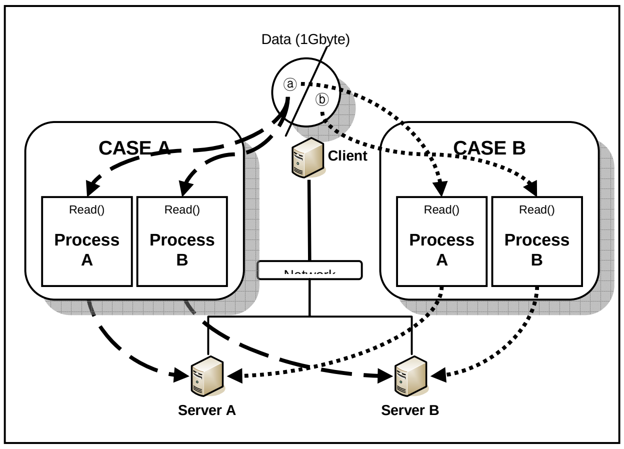


Figure B-1. Testbed for the Evaluation of the Disk IO Bottleneck

Table B-1. The Performance of the Hard Drive of Each Node

Source	Transfer Rate (MB/s)
Timing cached reads	673
Timing buffered disk reads	53

The evaluation consists of comparing the transfer time of each cases below.
- CASE A : split the data in two part(ⓐ,ⓑ), and send the same splited data part(ⓐ) to each server.
- CASE B : split the data in two part(ⓐ,ⓑ), and send each splited data part(ⓐ,ⓑ) to each server.

Through this evaluation we measured the performance degradation occurred when reading the different part of data simultaneously.

Table B-2 shows the result of this evaluation. As shown in this table, the transfer time of the CASE A is about 3.5 times faster than the transfer time of the CASE B.

Table B-2. Transfer Time for Transferring 500Mbytes of Bulk Data

Case	Transfer Time (second)
CASE A	24.2
CASE B	85.3

Therefore, when the network bandwidth is sufficiently large, the unicast transfer has a better performance than DDMG transfer (which is an extended mechanism of CASE B) because of the disk IO bottleneck.