GridFTP GUI: An Easy and Efficient Way to Transfer Data in Grid

Wantao Liu[1,2], Rajkumar Kettimuthu[3,4], Brian Tieman[5], Ravi Madduri[3,4], Bo Li[1], and Ian Foster[2,3,4]

[1] School of Computer Science and Engineering, Beihang University, Beijing, China
[2] Department of Computer Science, The University of Chicago, Chicago, IL USA
[3] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL USA
[4] Computation Institute, The University of Chicago, Chicago, IL USA
[5] Advanced Photon Source, Argonne National Laboratory, Argonne, IL USA
liuwt@uchicago.edu, kettimut@mcs.anl.gov, tieman@aps.anl.gov, madduri@mcs.anl.gov, libo@act.buaa.edu.cn, foster@mcs.anl.gov

Abstract. GridFTP is the de facto standard for providing secure, robust, high-speed bulk data transport. It is based on the Internet FTP protocol, and it defines extensions for high performance operation and security. However, GridFTP lacks an easy-to-use graphical user interface client that is fault tolerant and hides all the complexities from the end users. It is not straightforward to conduct data transfer, monitor transfer status, and recover from transfer errors. Many e-science applications must transfer large datasets that are, in many cases, are partitioned into lots of small files. However, existing GridFTP client tools cannot do such a transfer efficiently and reliably. To address these issues, we developed GridFTP GUI, a Java web start-based GridFTP client tool. It does not require explicit installation and can automatically update to the latest version. It provides an easy and efficient way for users to get credentials, transfer data through drag and drop, optimize transfer parameters, view transfer progress, move a lot of small files in an efficient way, recover from transfer errors, manage files and directories remotely, and establish cross-domain trust relationships. Our experiments show that GridFTP GUI is able to transfer files and directories with very good performance.

Keywords: GridFTP, data transfer.

1 Introduction

The GridFTP [1] protocol is widely used in Grid environments. It extends the standard FTP [2] protocol for high-performance operation and security. The Globus implementation of GridFTP [3] provides a modular and extensible data transfer system architecture suitable for wide-area and high-performance environments. GridFTP achieves good performance by using non-TCP protocols such as UDT [10] and parallel streams to minimize bottlenecks inherent in TCP [11]. GridFTP can also do coordinated data transfer by using multiple computer nodes at the source and destination. Globus GridFTP supports various security options, including Grid Security

T. Doulamis et al.: (Eds.): GridNets 2009, LNICST 25, pp. 57–66, 2010.

Infrastructure (GSI), username/password authentication provided by regular FTP servers, and SSH-based security.

Many scientific experiments produce hundreds of thousands of files to transfer every day. The individual file size is modest, typically on the order of kilobytes or megabytes, but the size of the entire dataset for each experiment is tremendous, ranging from hundreds of gigabytes to hundreds of terabytes. For example, tomography experiments on the Advanced Photon Source at Argonne National Laboratory produce hundreds of gigabytes of data every day. The datasets are typically organized into a hierarchical directory. The number of files under each subdirectory is moderate; however, the total number of files under a top-level directory that needs to be moved is huge. In a typical day, dozens of samples may be acquired; each sample generates about 2,000 raw data files. After processing, each sample produces additional 2,000 reconstructed files; each file is 8 to 16 MB in size.

Many scientists use GridFTP to transfer their experimental data from sites where experiments are conducted to computational sites where the acquired data is analyzed. After the data has been processed, the output data must be archived at other sites. Globus-url-copy is a commonly used GridFTP client. Since it is a command-line tool, typically scripts are used to perform the cited tasks. Because discipline scientists often have only limited computer knowledge and experience, dealing with scripts and command-line interfaces is difficult and tedious. Contending with cross-platform issues, tuning performance parameters, recovering from transfer errors, and managing files and directories in remote GridFTP servers make the process even more difficult. The scientific research user communities need a tool for GridFTP data transfer and management that provides the following features:

1. Ease of use: It should not take much time to learn how to use this tool. A graphical user interface therefore is preferable to a command-line interface.
2. Monitoring: Users should be able to monitor the status of the transfers.
3. Remote file management: Users should be able to manage the remote files and directories by performing operations such as create/delete remote files and directories.
4. Performance tuning: The tool should support automated and manual tuning of parameters to improve performance.
5. Error recovery: The tool should be able to recover from any failures and restart transfers automatically.
6. Efficiency: Data movement activity is an integral part of the scientific experiments, a significant part in many cases. If the tool is not efficient enough, it will impact the whole experiment significantly.

GridFTP GUI, a cross-platform client tool for GridFTP, has these six features. It has a simple and familiar interface; hence, users do not take long to learn how to use it. GridFTP GUI is based on Java web start technology [4]. Users can get it by clicking a link in a web page; the program will be downloaded and started automatically. There is no explicit installation process.

The paper is organized as follows. In Section 2, we review related work. Section 3 presents the design and implementation of GridFTP GUI. In Section 4, we discuss experimental results. Section 5 presents conclusions and future work.

2 Related Work

RFT (Reliable Transfer Service) [8] is a component of the Globus Toolkit. Implemented as a set of Grid services, RFT performs third-party transfers using GridFTP with basic reliable mechanisms. Data transfer state is dumped into database; if a transfer fails, it can be restarted automatically from the broken point by using the persistent data. RFT offers a command-line client for end users to conduct reliable third-party transfers. GridFTP GUI is a lightweight client that supports both client-server and third-party GridFTP transfers. Also, the GridFTP GUI has been integrated with RFT to provide better fault-tolerant capability for third-party transfers.

GridCopy [12] is a command-line client for GridFTP. It accepts SCP-style source and destination specifications. GridCopy translates the SCP-style pathnames into appropriate GridFTP URLs. Moreover, it is able to calculate an optimal value of TCP buffer size and parallel TCP streams for the GridFTP transfer to maximize throughput. But it does not provide an easy way to manage remote files. Also, GridFTP GUI provides better fault-tolerant capability than does GridCopy.

Topaz [5] is an extension of the Firefox browser to enable GridFTP capability. This work is based on Mozilla framework. However, because of the limitations of the browser, Topaz implements only file download and upload. Other features of GridFTP are not supported; remote file management is not supported as well.

UberFTP [6] is an interactive shell GridFTP client. It supports GSI authentication, parallel data channels and third party transfer. UberFTP is not able to conduct reliable file transfer, however, whereas GridFTP GUI incorporates with RFT service to offer reliability in file transfers.

GridFTP Client [7] is an eclipse-based Java application. It is distributed as a standalone application or an eclipse plug-in. It is able to transfer a directory that includes a lot of nested subdirectories, create and delete remote files and directories, monitor transfer status. However, GridFTP Client supports only third-party transfers; thus, users cannot download or upload files between GridFTP server and GridFTP Client.

The CoG jglobus [13] library includes a pure Java GridFTP client API. It can be used to build applications that communicate with GridFTP servers. It supports client-server transfers, third-party transfers, parallel streams and TCP buffer size setting, striping, GSI authentication, and other features. GridFTP GUI uses CoG jglobus to communicate with GridFTP servers.

3 GridFTP GUI Design and Implementation

The heterogeneity of Grid environments is reflected not only in the server side but also in the client side. Users requesting Grid resources run on a great diversity of platforms. Hence, cross-platform issues must be taken into account when designing and implementing Grid client software. For the GUI to operate on a variety of platforms, we have developed the GUI in the Java programming language.

Since our aim is to enable the average user easily to perform GridFTP transfers, we adopted the Java web start technology. With Java web start, the user can access the application with a single click. The application is then downloaded to user's client machine and installed to a system-dependent location for the first time launch. For

subsequent uses, the user can just invoke the application on his machine. Java web start automatically checks whether the application has a new version in the software repository; if so, it downloads the new version without user intervention. Users can always get the most recent version of the application without any manual installation.

Obtaining certificates and setting up GSI are difficult tasks for an average user. The GridFTP GUI has been integrated with MyProxy [9] to reduce the burden of managing certificates. MyProxy is an X509 PKI credential management service that enables users to upload their credentials to an online credential repository and obtain them when needed. MyProxy also can act as a certificate authority (CA). Virtual organizations can use its CA capability to issue certificates to their users. It can be configured in such a way that the users can obtain the certificates by typing in a username and password. With GridFTP GUI, users do not need to explicitly obtain or handle the certificate from the MyProxy server. Through the GUI, users can authenticate with the MyProxy server using a username and password and let the GUI handle the retrieval and setting up of the certificates.

3.1 Design of GridFTP GUI

GridFTP GUI is modular. Its architecture is presented in Figure 1. The topmost layer is an assembly of graphical user interface elements; specifically, they are Java Swing controls, which are responsible for interaction with users. This layer is divided into several parts in terms of functionality:

- *Cred Manager* is used for credential management. It interacts with MyProxy and local GSI credential files. Users can generate and destroy credentials and view credential information.
- *Login Manager* accepts user specified GridFTP server name and port, constructs commands that are used for GridFTP login and pass them to command executor.
- *File Browser* controls how local and remote files are displayed; tree view is implemented currently. It supports navigation among different file hierarchies. Users can create directories and delete or move files or directories through file browser. Moreover, users can drag and drop files or directories among file browsers to trigger GridFTP data transfer.
- *Status Manager* provides a view for users to see the status of completed transfers and the progress of on-going transfers.
- *Config Manager* is responsible for configuration of GridFTP parameters, such as the TCP buffer size or parallel streams, via a graphical interface.
- *Log Manager* allows the user *to* set the log file location and views log file.

Figure 2 shows a snapshot of GridFTP GUI. The command executor component, receives user commands passed from the upper-layer GUI elements and executes them. Commands are divided into two categories: control commands and transfer commands. Control commands control the behavior of GridFTP GUI (e.g., open a file browser, generate a new proxy) and do not interact with the GridFTP server. The command executor executes these commands immediately for timely response. Transfer commands instruct the GridFTP server or RFT service to finish some tasks; they

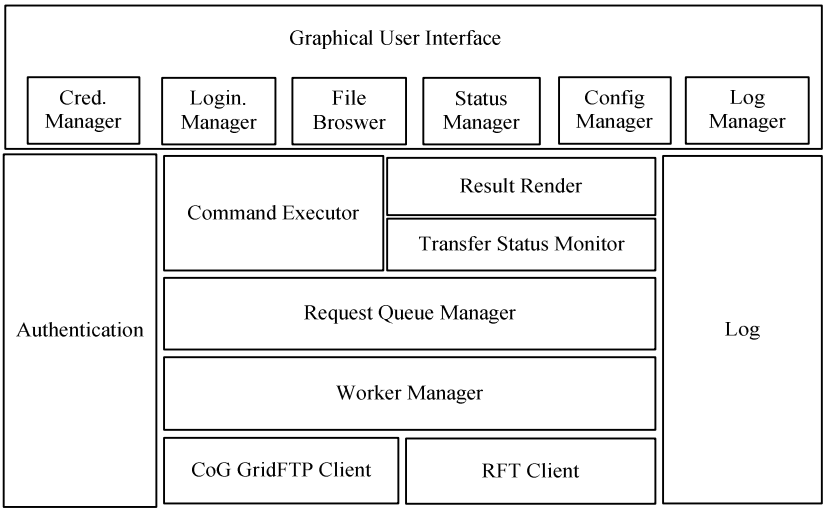

Fig. 1. Components of GridFTP GUI

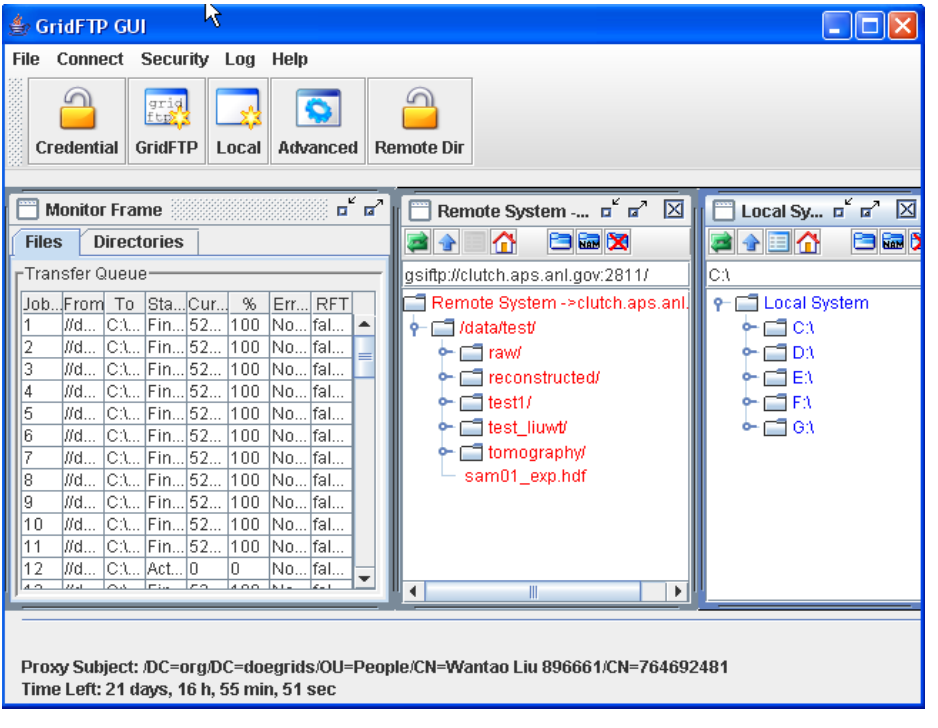

Fig. 2. Snapshot of GridFTP GUI

are put into a queue managed by the request queue manager. Since there are two types of commands, the command executor is designed as an abstract interface, which has different concrete implementations to handle corresponding types of commands. This design facilitates the extensibility of the program.

The request queue manager uses a specific policy for queuing transfer requests. Currently, only the first come first serve (FCFS) policy is implemented. All requests in the request queue are processed by worker threads in a thread pool. These threads are managed by the worker manager, which assigns requests for different workers and monitors the execution of these threads. According to different commands, worker threads invoke the CoG jglobus client or RFT client for data transfer.

The transfer status monitor is used for checking transfer status and progress. The status information is then passed to the result renderer, which can display this information in different ways.

3.2 Handling Large Numbers of Files

Moving a lot of files from one location to another is a requirement in e-science. To reach high-throughput levels, conventional GridFTP clients require not only that the amount of data to transfer be large enough to allow TCP to reach full throttle but also that the data be in large files, ideally in one single file. If the dataset is large but partitioned into many small files (on gigabit networks we consider any file smaller than 100 MB as a small file), the performance suffers drastically. To address this issue, we use a pool of transfer worker threads to handle the request queue. Multiple files are moved concurrently to maximize the utilization of network bandwidth and thus improve the transfer performance.

Displaying transfer status clearly and methodically in the GUI when moving a lot of files is a challenging task as well. The GUI needs to provide users well-organized information so that the users can easily view the status of the transfer. Based on these considerations, we use two Java Swing JPanels to show transfer information. One JPanel lists all directories and their transfer status; the other JPanel lists all files under the active directory. This solution offers users an easy way to know the overall transfer status and progress of a specific file.

When the program starts moving a directory that contains subdirectories and a lot of files, the directory is traversed recursively, and only nested subdirectory nodes are returned and put into a queue. All these nested subdirectory nodes are added to the directory transfer panel. After the traversal, queued subdirectory nodes are tackled in FCFS order. For a subdirectory node, all the regular files under the subdirectory are retrieved and added to the file transfer request panel. After retrieval of a subdirectory, a corresponding transfer request is constructed for each regular file and put into the request queue. When a transfer request status changes, the file transfer request panel updates the corresponding entry. Since the request queue size is limited, when the queue is full, the thread that is putting requests into the queue is suspended until there is space again.

We retrieve all regular files of a subdirectory only after all the files in the previous subdirectory are transferred. Thus, significant delay could occur between the retrieval of the file list in one subdirectory and the retrieval of the file list in the next subdirectory. Hence, the GridFTP control connection could time out. To keep the control

connection alive, we implemented a thread that periodically sends a NOOP command to the GridFTP server.

3.3 Error Recovery

Errors can occur during data transfer for various reasons, such as disk failure, network outage, and end system problems. While transferring a lot of individual files or a directory with lots of files, it would be tedious to have to manually identify and re-transfer the files that had not been transferred successfully. Therefore, automatic recovery from errors is an important feature.

GridFTP GUI supports automatic error recovery capabilities. The GUI will retry any failed transfers. The number of retries and the interval between subsequent retries can be configured. Also, the GUI stores the necessary status information on the local file system so that the transfer can be resumed seamlessly after a local failure, such as an unexpected GUI crash or client machine failure. The GUI has also been integrated with RFT to provide more robust fault-tolerant capabilities. RFT stores the transfer state in a persistent store.

3.4 Establishment of Cross-Domain Trust Relationships

In Grid environments, data commonly is transferred among participants located in different organizations, countries, or even continents, each having individual policies and trust certificates issued by distinct accredited authorities. Hence, establishing trust relationships is a big challenge for users conducting cross-domain communication.

The International Grid Trust Federation (IGTF) [14] is an organization that federates policy management authorities all over the world, with the goal of enhancing establishment of cross-domain trust relationships between Grid participants. The distribution provided by IGTF contains root certificates, certificate revocation list locations, contact information, and signing of policies. Users can download this distribution and install it to conduct cross-domain communication.

GridFTP GUI interacts with IGTF. When GridFTP GUI starts up, it contacts IGTF's website and checks the available distribution to see whether there is any update. The latest distribution will be downloaded and installed into the directory where the user puts the certificates. This feature simplifies the user's effort in establishing cross-domain trust relationships with other Grid entities.

4 Performance Evaluation

In this section, we present some performance evaluation results of GridFTP GUI compared with scp and globus-url-copy. Scp [15] is a widely used command line tool for remote data transfer with security. It is able to copy files between hosts on a network and provides the same security as ssh. Globus-url-copy [16] is a command line GridFTP client shipped with Globus Toolkit. Globus-url-copy supports multiple parallel data streams for a single transfer; this feature can dramatically improve performance when transferring a big file. This optimization parameter is used in our experiment. Because of the broad acceptance of the two tools, they are selected to compare with GridFTP GUI.

We conducted all of our experiments using TeraGrid NCSA nodes and the University of Chicago nodes. Each experiment was run three times, and the average value is presented here. The GridFTP servers involved in the experiments were configured to use TCP as the underlying data transport protocol; scp uses TCP as well. As stated above, globus-url-copy is able to establish multiple data streams to transfer a single file; in the following charts, "globus-url-copy(p=4)" refers to four data streams for a single file transfer. Besides parallel streams, GridFTP GUI can transfer multiple different files simultaneously through its transfer worker thread pool; the thread pool size was set to four in all experiments. "GridFTP GUI(p=4)" denotes each worker thread has four data streams, and "GridFTP GUI" denotes each worker thread has only one data stream.

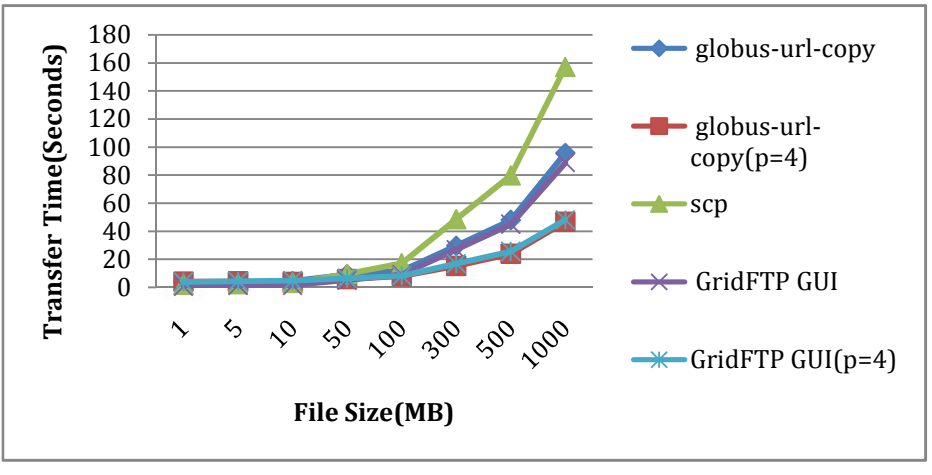

Fig. 3. Data transfer time of a single file (from NCSA to UChicago)

In the first experiment, we examined the time consumed to transfer a single file. In this experiment, after a file transfer was finished, we requested another file transfer. Eight file sizes were used: 1 MB, 5 MB, 10 MB, 50 MB, 100 MB, 300 MB, 500 MB, and 1000 MB. Data was moved from NCSA to the University of Chicago. Figure 3 shows the results of this experiment. We can see that the five curves are very close when the data size is less than 10 MB. After that, both globus-url-copy and GridFTP GUI with four data streams outperforms other programs remarkably. It should be noted that GridFTP GUI's performance is as good as globus-url-copy. Though the default number of worker threads is set to four, only one thread is used here as this experiment involved only single file transfers.

Figure 4 and 5 demonstrate performance for directory. In the two experiments, only globus-url-copy with four data streams and GridFTP GUI with one data stream for each worker thread are examined (in Figure 4, scp is compared as well). Figure 4 shows the data transfer time of a directory with tens of thousands of small files. It is a simulation of a typical scenario in e-science. In this evaluation, we created directories of 10,000, 20,000, 30,000, 40,000 and 50,000 files. Each file was 1 MB. Therefore, the total data size moved ranged from 10 GB to 50 GB. Data flowed from the

University of Chicago to NCSA. From the plot, we can see that GridFTP GUI remarkably outperforms scp and globus-url-copy with four streams. The multiple concurrent transfer threads in GUI help it achieve superior performance while transferring directories with lots of small files. Globus-url-copy can use multiple concurrent transfers to achieve performance similar to that of GUI. Still, GUI will be much more easier to use for the scientific community than globus-url-copy.

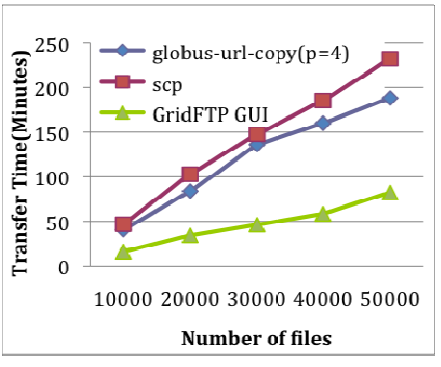

Fig. 4. Transfer of directory containing lots of 1 MB files (from UChicago to NCSA)

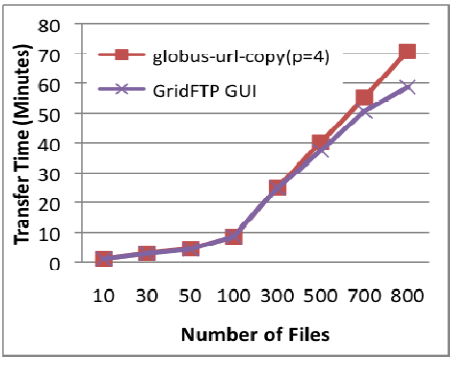

Fig. 5. Transfer of directory containing 100 MB files (from UChicago to NCSA)

To investigate the performance of GridFTP GUI when moving a directory with large files, we carried out another experiment. A directory with different number of files was moved from the University of Chicago to NCSA; each file was 100 MB. The number of files in a directory ranged from 10 to 800. As depicted in Fig. 5, GridFTP GUI shows moderate improvement compared with globus-url-copy with four parallel streams.

5 Conclusions and Future Work

In this paper, we presented GridFTP GUI, a comprehensive GridFTP client that enables using users to generate a proxy certificate or obtain it from MyProxy, conduct client/server or third-party data transfers by drag and drop, view transfer progress, set GridFTP server parameters, manage files and directories in remote GridFTP servers, and recover from transfer errors automatically. GridFTP GUI is a Java web start application; hence, no explicit installation process is required, and the program can automatically update to the latest version.

We plan to implement more sophisticated ways of managing transfer requests. In the current implementation, all transfer requests are put into the queue in FCFS order. Since priority of data transfer request is important in some cases, however, we are going to add more sophisticated queuing policies to support priority. Another enhancement we intend to make is to automatically optimize transfer parameters. Configuring transfer parameter is not easy for users, and the optimal value of those parameters is affected by multiple factors. Hence, it will be helpful if GridFTP GUI

can automatically optimize transfer parameters. In [12], a practical approach for GridFTP transfer parameter estimation is proposed. We plan to use that approach as a starting point to provide this capability.

Acknowledgments

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

1. Allcock, W.: GridFTP: Protocol Extension to FTP for the Grid. In: Global Grid Forum GFDR- P.020 (2003)
2. Postel, J., Reynolds, J.: File Transfer Protocol. IETF, RFC 959 (1985)
3. Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., Foster, I.: The Globus Striped GridFTP Framework and Server. In: SC 2005. ACM Press, New York (2005)
4. Java Web Start, http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp
5. Zamudio, R., Catarino, D., Taufer, M., Bhatia, K., Stearn, B.: Topaz:Extending Firefox to Accommodate the GridFTP Protocol. In: Proceedings of the Fourth High-Performance Grid Computing Workshop HPGC 2007, in conjunction with IPDPS 2007, Long Beach, California (March 2007)
6. Uberftp, http://dims.ncsa.uiuc.edu/set/uberftp/
7. GridFTP Client, http://bi.offis.de/gridftp/downloads.html
8. RFT, http://globus.org/toolkit/docs/latest-able/data/rft/#rft
9. MyProxy, http://grid.ncsa.uiuc.edu/myproxy/
10. Gu, Y., Grossman, R.L.: UDT: UDP-based Data Transfer for High-Speed Wide Area Networks. Comput. Networks 51(7), 1777–1799 (2007)
11. Postel, J.: Transmission Control Protocol. IETF, RFT 793 (September 1981)
12. Kettimuthu, R., Allcock, W., Liming, L., Navarro, J., Foster, I.: GridCopy: Moving Data Fast on the Grid. In: Fourth High Performance Grid Computing Workshop HPGC 2007, in conjunction with IPDPS 2007, Long Beach, California (March 2007)
13. CoG jglobus, http://dev.globus.org/wiki/CoG_jglobus
14. IGTF, http://www.igtf.net/
15. Scp, http://en.wikipedia.org/wiki/Secure_copy
16. GridFTP, http://www.globus.org/toolkit/docs/4.0/data/gridftp/rn01re01.html