Evaluating and Enhancing the Use of the GridFTP Protocol for Efficient Data Transfer on the Grid

Mario Cannataro[1], Carlo Mastroianni[2], Domenico Talia[3], Paolo Trunfio[3]

[1] University “Magna Græcia” of Catanzaro, Via T. Campanella 115,
88100 Catanzaro, Italy, cannataro@unicz.it

[2] ICAR-CNR, Via P. Bucci 41/c,
87036 Rende, Italy, mastroianni@icar.cnr.it

[3] University of Calabria, Via P. Bucci 41/c,
87036 Rende, Italy, {talia, trunfio}@deis.unical.it

Abstract. Grid applications often require large data transfers along heterogeneous networks having different latencies and bandwidths, therefore efficient support for data transfer is a key issue in Grid computing. The paper presents a performance evaluation of the GridFTP protocol along some typical network scenarios, giving indications and rules of thumb useful to select the “best” GridFTP parameters. Following some recent approaches that make use of experimental results to optimize data transfers, the paper presents a simple algorithm that suggests the “best” GridFTP parameters for a required transfer session on the basis of historical file transfer data.

1 Introduction

The *Grid* is an integrated infrastructure for coordinated resource sharing and problem solving in distributed computing environments. Grid applications often involve large amounts of data and/or computing, therefore efficient support for data transfer on networks with different latencies and bandwidths is a key issue.

GridFTP [1] is a protocol, developed within the context of the Globus Toolkit, that supports the efficient transfer of large amounts of data on Grids, facing high latency and low bandwidth problems often encountered in geographical networks.

GridFTP is based on the FTP protocol but, opposite to many implementations of that protocol, it supports and extends a large subset of the features defined in the RFC 969 standard [2]. The major contribution of the GridFTP protocol is the use of the security services of Globus to assure the authentication of Grid applications. Other peculiar features of GridFTP are: the manual setting of the TCP buffer size, the use of multiple parallel streams, the third-party transfer option, and the partial file transfer option.

Some recent works report performance evaluations of the GridFTP protocol or propose techniques that make use of experiment results to optimize data transfers. In [3], GridFTP is compared with the Remote File I/O protocol developed at CERN, and it is shown that GridFTP is generally more efficient if TCP buffer size and the number of sockets are properly tuned. In [4], a client library that provides a set of high level functionalities based on GridFTP and other basic Globus services, is proposed. Some performance evaluations are also reported for file transfers between Italy, UK and USA. In [5], the problem of efficient data transfer is tackled within the context of the Data Grid project. The goal was to predict future data transfer performances by using statistical functions calculated over past experiments. All file transfers use 8 parallel streams and a TCP buffer size equal to the theoretical optimum value (Bandwidth times Round Trip Time). In [6], a method is proposed to combine active tests (i.e., when test data are sent through the network) and passive tests (i.e., when useful data are transferred and results are used as test results). That work proposes to publish summarized test results on the Grid information services, in particular on the Monitoring and Discovery Service of Globus, to make them accessible by Grid hosts. Hosts are therefore aware of end-to-end connection properties before starting a file transfer.

This paper discusses a work that follows the approach suggested by [5] and [6]. After a performance evaluation of the GridFTP protocol along some typical network scenarios, the paper describes a tool for the collection and summarization of GridFTP usage data (collected for each transfer session activated on a node), that implements a simple procedure whose purpose is to suggest the “best” GridFTP parameters for a required transfer session.

The rest of the paper is organized as follows. Section 2 presents the performance evaluation of GridFTP through different data transfer scenarios and summarizes the main results obtained and some possible indications on how to choose the GridFTP parameters. Section 3 presents a tool that automatically configures the GridFTP parameters to minimize file transfer times between Globus hosts. Finally, Section 4 concludes the paper.

2 GridFTP performance evaluation

In order to evaluate the performance of the GridFTP protocol in different network scenarios, we tested GridFTP along three different kinds of connections with the following tests:

1. Tests on adjacent Local Area Networks (LAN). Files were transferred between two hosts belonging to different LANs connected through a router.
2. Tests on two different long distance Internet connections; we chose connections with similar bandwidth characteristics but different latency values.

The `pipechar` tool [7] was used to evaluate the main characteristics of the connections, that is the RTT (Round Trip Time) delay and the bottleneck bandwidth (i.e., the minimum bandwidth measured on the path followed by IP packets).

Transfers were executed during the night to avoid network congestion, and, more important for the objective of this work, to minimize the effect of the network load variability during the experiments. The low variance of the measurements we obtained shows that this goal has been achieved.

We used the `globus-url-copy` command of the Globus Toolkit version 2.2, with the following syntax:

```
globus-url-copy -vb -notpt -tcp-bs <buffer> \
-p <parallel> gsiftp://<source-file> gsiftp://<dest-file>.
```

Where `<buffer>` is the TCP buffer size (in Kbytes) and `<parallel>` is the number of parallel streams (i.e., the number of sockets used in parallel for the transfers).

The following parameters have been varied in the tests:

- the TCP buffers on sender and receiver, with values from 1 Kbyte to 64 Kbytes, that is the maximum allowed size on many operating systems;
- the number of parallel streams: tests were made using 1, 2, 4, 8, 16, 32, and 64 sockets;
- the file size, with values from 32 Kbytes to 64 Mbytes.

For each combination of these parameters, we performed 20 file transfers and calculated the average data transfer value after discarding values that differ more than 20% with respect to the overall average.

2.1 Tests between adjacent LANs

These tests have been performed between a host (`telesio.cs.icar.cnr.it`) at ICAR-CNR (Institute for High Performance Networks and Applications) and a host (`griso.deis.unical.it`) at University of Calabria. These hosts belong to 100 Mbps LANs connected through 10 Mbps links to a router.

Preliminary tests run with `pipechar` showed that for this connection the mean RTT delay is 31.5 msec, while the bottleneck bandwidth is 4.7 Mbps, with a theoretical optimum TCP buffer size equal to 18.5 Kbytes (TCP buffer size = RTT $\times$ bandwidth).

File transfer experiments confirmed the presence of an optimum TCP buffer, even if its size is lower than the theoretical one. Figure 1a reports the data transfer rates obtained with the transfers of a 16 Mbytes file, w.r.t. the TCP buffer size, for different value of the number of sockets. Figure 1b shows the same performances w.r.t. the number of sockets, to highlight the effect of using several sockets. From those figures two considerations arise:

(i) a high number of sockets is not advantageous for a high bandwidth connection like this, because the amount of time needed to set up and release the connections outweighs the possible advantage of having many parallel streams; the use of a small number of sockets (from 4 to 8) seems to be a good choice for all the values of the TCP buffer size;

(ii) the buffer size that gives the best performance decreases as the number of sockets increases: while with 1 or 2 sockets a 8 Kbyte buffer is the best choice, with 4, 8 or 16 it is better to use a 4 Kbyte buffer, and an even smaller buffer (2 Kbytes) is preferable if 32 sockets are used. A motivation can be that, with a high number of sockets, the operating system is requested to manage a large amount of TCP buffer memory: e.g., with 32 sockets and a 64 Kbyte buffer, the overall buffer memory is equal to 2 Mbytes.

Similar qualitative results were obtained for transfers of larger files. Figure 2 reports a summary of data transfer rates for different file sizes, and confirms that the use of several sockets is not effective for this kind of connection. We also can note that, as expected, the transfer data rate increases when the file size increases, due to the lower relative impact of connection set up and release phases. Figure 2 also shows that it is not necessary to test transfers of very large files, since curves tend to get to a saturation.

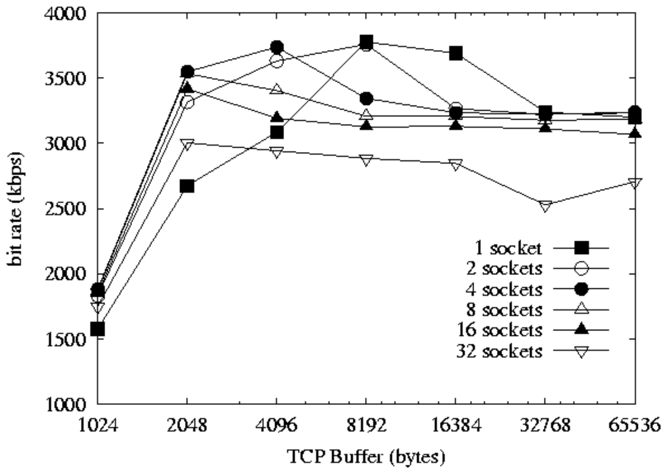

Fig. 1a. Data transfer rates for a 16 Mbyte file, versus the TCP buffer, for different values of the number of parallel streams.

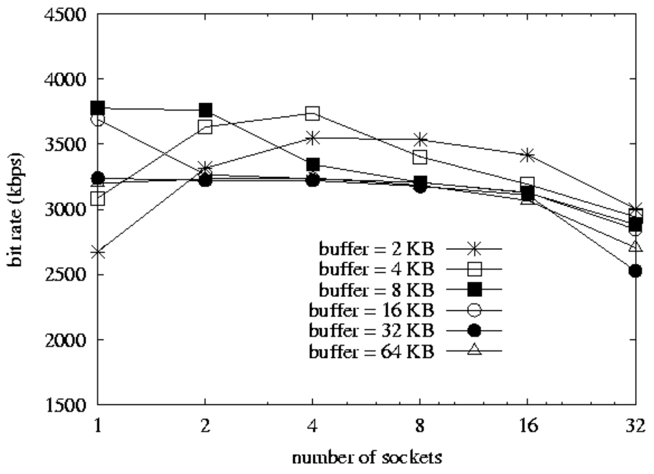

Fig. 1b. Data transfer rates for a 16 Mbyte file, versus the number of sockets, for different values of the TCP buffer size.

2.2 Tests on Internet connections – case A

GridFTP tests were also made from an ICAR-CNR host (`icarus.cs.icar.cnr.it`) to a host at the University of Calabria (`griso.deis.unical.it`). Differently from the case analyzed in Section 2.1, routers were configured so that the LANs were not directly linked, but IP packets followed a path along the Italian high-bandwidth GARR network (with a number of hops equal to 8). The resulting bottleneck bandwidth is about 1.6 Mbps, while the mean RTT delay is 80 milliseconds, with a theoretical optimum TCP buffer size equal to about 16 Kbytes.

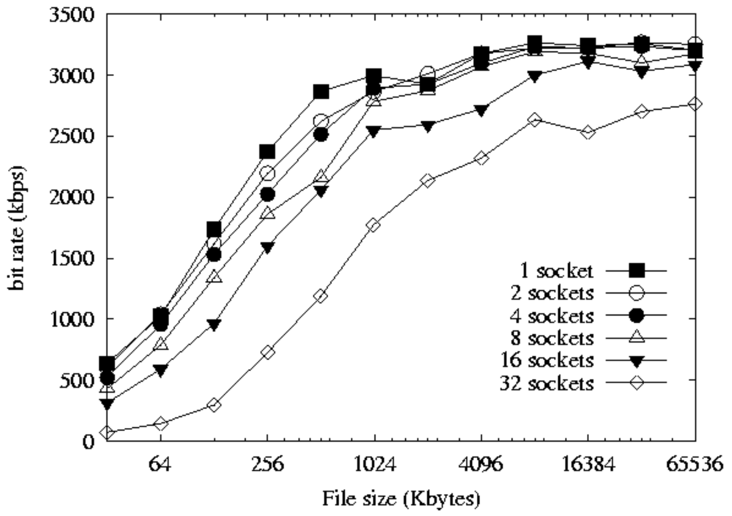

Fig. 2. Data transfer rates versus the file size, for different values of the number of sockets. The TCP buffer size is set to 32 Kbytes.

Figure 3a depicts data transfer rates obtained with a 16 Mbyte file. It appears that when the number of sockets increases, the larger potential transfer rate is balanced by the longer connection procedures: a good trade-off seems to be reached when the number of streams is between 16 and 32.

For what concerns the TCP buffer size, the most convenient size strongly depends on the number of sockets: with only 1 socket, a 64 Kbyte buffer is to be chosen, while with 32 or 64 sockets a 8 Kbyte buffer gives the best performance. With 8 or 16 sockets the performance obtained with different buffer sizes is similar. As a consequence, we may note that a non optimal choice of the buffer size would not cause a notable performance degradation.

Figure 3b shows the performance obtained transferring a 64 Mbyte file. The larger file size causes two remarkable phenomena: (i) performance does not worsen when the number of sockets increases, and (ii) the buffer size influence is very small with a large number of sockets. Therefore, it results that transfers of 64 Mbytes or larger files should be made with 32 or 64 sockets, while the choice of the buffer size is almost ineffective.

Figure 4 shows that a number of sockets ranging from 4 to 8 gives the best performance, w.r.t. to lower or larger numbers of sockets, with almost the considered file sizes. When using a higher number of sockets (e.g., 32), we also may obtain good performance if the file size exceeds 32 Mbytes, but experiment poorer performance when transferring smaller files.

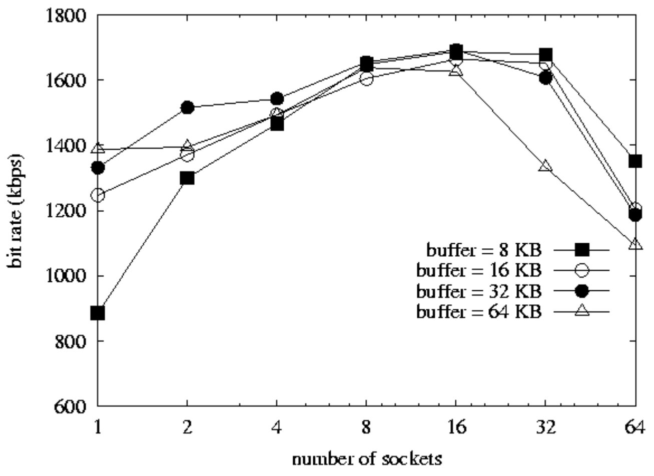

Fig. 3a. Data transfer rates for a 16 Mbyte file, versus the number of sockets, for different values of the TCP buffer size.

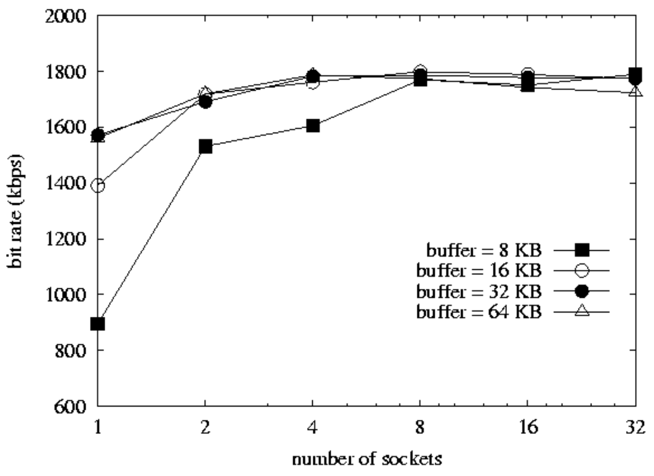

Fig. 3b. Data transfer rates for a 64 Mbyte file, versus the number of sockets, for different values of the TCP buffer size.

2.3 Tests on Internet connections – case B

These tests were performed between an ICAR-CNR host (`icarus.cs.icar.cnr.it`) and a host at the CNUCE-CNR institute in Pisa (`novello.cnuce.cnr.it`). The path between these two nodes includes both inter-LAN connections and Internet links. With respect to the connection discussed in Section 2.2, this connection has a similar bottleneck bandwidth (about 1.7 Mbps), but a higher RTT delay (125 msec). These parameters lead to a theoretical optimum TCP buffer equal to about 26.5 Kbytes.

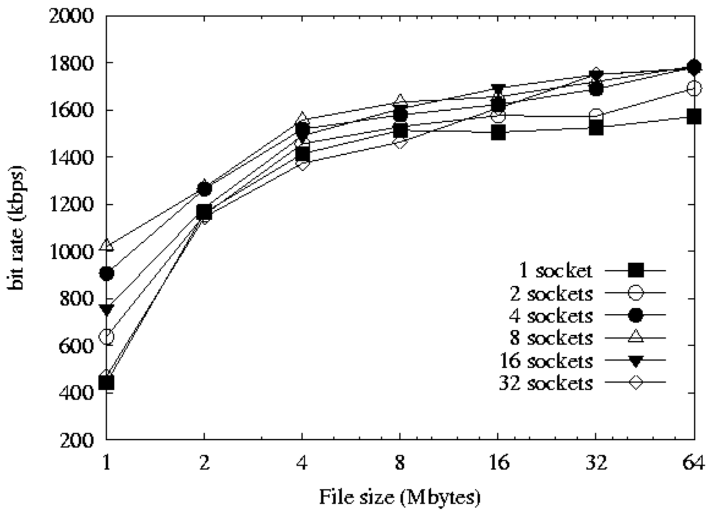

Fig. 4. Data transfer rates versus the file size, for different values of the number of sockets. The TCP buffer size is set to 32 Kbytes.

We experimented that, due to the higher latency, performance figures show some remarkable differences when they are compared to the figures reported in Section 2.2, though the maximum data rates that are achievable with both connections are similar.

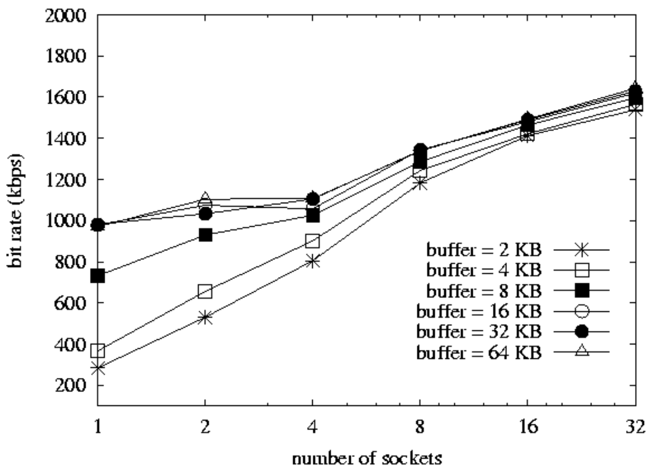

Fig. 5. Data transfer rates for a 16 Mbyte file, versus the number of sockets, for different values of the TCP buffer size.

In Figure 5 we see that, when transferring a 16 Mbyte file, performance increases with the number of sockets for all values of the TCP buffer. Furthermore, a high TCP buffer is advantageous with any number of sockets, though the advantage decreases as that number increases. Therefore, with this file size, the best choice is to have a number of sockets and a TCP buffer size as large as possible.

This is not true for all file sizes. Figure 6 shows performances w.r.t. the file size, with a 32 Kbyte TCP buffer: we see that a high number of sockets is beneficial only for big-sized files, while for small files 1 or 2 sockets are preferable. Note that curves related to different numbers of sockets get crossed in the range between 512 Kbytes and 2 Mbytes.

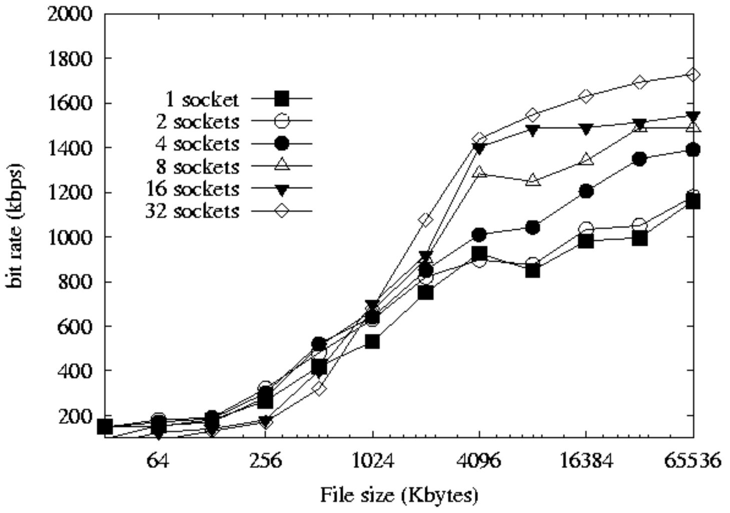

Fig. 6. Data transfer rates versus the file size, for different values of the number of sockets. The TCP buffer size is set to 32 Kbytes.

2.4 Summary results

On the basis of the experiments presented along the paper, we can draw some conclusions on performance trends obtained by varying the file size, the TCP buffer and the number of sockets:

1. The optimal number of sockets strongly depends on the type of connection, particularly on the latency value: for low-latency connections, a small number of sockets is sufficient, while as the latency increases the use of more sockets can lead to a significant advantage;
2. The file size has also an impact on the choice of the number of sockets: a high number of sockets becomes convenient only for files whose sizes exceed a certain value. This cross value depends on the Grid connection and decreases as the latency increases: for example for the Internet connection reported in Section 2.2 (lower latency), the cross value is about 32 Mbytes, while for the connection reported in Section 2.3 (higher latency), the cross value is about 2 Mbytes.
3. The optimal TCP buffer size depends on the network connection (latency and bandwidth), the file size, and the number of sockets. Dependencies are complex

but a rule of thumb can be the following: large TCP buffers are advantageous with high RTT × BW products and for transfers of large files; but the use of a high number of sockets often requires small TCP buffers.

In summary, supposing to know the network characteristics (latency and bandwidth) and given the file size, a possible approach to choose the GridFTP parameters could start finding the number of sockets, on the basis of latency and file size, and then finding the TCP buffer size.

3 A tool for enhanced GridFTP file transfers

In Section 2 we reported some of the performance results we obtained with GridFTP tests on three different types of network connections.

At completion of this work, we built a tool, written in Java and executable on machines running the Globus Toolkit, that uses the GridFTP protocol to perform efficient file transfer on a Globus-based Grid.

Such tool has two main goals:

- *to build a file transfer log*, by collecting the performance data about the executed file transfers;
- *to enhance the file transfer*, by automatically setting the GridFTP parameters (TCP buffer and number of sockets) to those values that are supposed to minimize the transfer time for a given file on a given Grid connection.

The automatic setup of the GridFTP parameters is made on the basis of the experience, i.e., of the transfer times obtained in previous file transfers and stored in the file transfer log.

In particular, when a file transfer between hosts *A* and *B* is requested, the following steps are executed:

(i) the tool tries to determine the type of Grid connection, in order to select the historical data that can be useful. For example, if nodes *A* and *B* belong to Internet domains for which historical data are already available, the tool will refer to those data. The connection type can also be derived by measurements on network parameters (RTT, BW) made with the `pipechar` tool. If data are available for another connection with similar parameter values, those data can be referred. If none of those cases applies, the tool generates a new connection type and suggests the user to perform a set of tests that can be used in the future.

(ii) once the reference historical data have been chosen, the tool determines the GridFTP parameter values that are the most convenient on the basis of the experience.

(iii) the file transfer is executed with the chosen parameters, and the transfer times are used to update the historical data for the connection type under consideration.

To transfer a very large file, it can be convenient to split the file in several parts and separately transfer those parts. In this way, GridFTP parameters could be adapted on the fly to network conditions that may vary between transfers of file parts.

The tool provides a graphical interface allowing a user to navigate in the file systems of the Grid hosts and to execute transfers of one or more files with intuitive drag and drop operations. Before starting the transfer, the tool proposes to the user the convenient parameters (that she/he can modify) and shows the estimated transfer time. Moreover, the tool automatically manages the Globus authentication procedures needed prior to start the GridFTP transfers.

4. Conclusions and Future Work

In this paper we discussed a performance evaluation of the Globus GridFTP protocol along some typical network scenarios. We obtained some indications and rules of thumb useful to choice the GridFTP parameters if main network characteristics are known. The high variability of network conditions makes it hard to find analytical, close formulas to find such parameters.

Following some recent approaches that make use of experimental results to optimize data transfers, we built a Java tool executable on Globus-based machines, whose objective is to allow the user to perform efficient data transfers on the Grid by means of a user-friendly graphical interface. Such tool suggests the “best” GridFTP parameters for a required transfer session on the basis of historical data. Currently, the finding of stored historical data exploitable for the requested file transfer to be optimized is obtained through a simple similarity function. However, we are designing a tool extension that will make use of data mining techniques, such as clustering and classification techniques, that will produce a categorization of Grid connections and will allow a more effective selection of historical data exploitable for the current file transfer. In the next future we will offer the file transfer tool to interested users through the Web.

Acknowledgments

We acknowledge the CNUCE-CNR Institute (now ISTI-CNR), for kindly providing the access to their Grid hosts and allowing the execution of our GridFTP tests.

This work has been partially funded by the project ”MIUR Fondo Speciale SP3: GRID COMPUTING: Tecnologie abilitanti e applicazioni per eScience”.

References

1. The Globus Project: the GridFTP protocol, http://www.globus.org/datagrid/gridftp.html
2. RFC 969: NETBLT: A Bulk Data Transfer Protocol, http://www.faqs.org/rfcs/rfc969.html
3. Kalmady, R., Tierney, B.: A Comparison of GSIFTP and RFIO on a WAN. Technical Report for the Work Package 2 of the European DataGrid project, http://edg-wp2.web.cern.ch/edg-wp2/publications.html (2001)
4. Aloisio, G., Cafaro, M., Epicoco, I.: Early experiences with the GridFTP protocol using the GRB-GSIFTP library. Future Generation Computer Systems 18 (2002)
5. Vazhkudai, S., Schopf, J. M., Foster, I.: Predicting the Performance of Wide Area Data Transfers. Proc. International Parallel and Distributed Processing Symposium (2002)
6. The European Datagrid Project: DataGrid Network Monitoring Scheme Proposal, document DataGrid-07-TED-nnnn-0_1 (2001)
7. The Pipechar Tool: http://www-didc.lbl.gov/pipechar