GridFTP Update January 2002

This document is intended to summarize the current status of GridFTP and to address recurring issues. It will not be updated on any regular basis, but as needed.

The document provides a brief technical summary of GridFTP and then addresses the following topics:

- What GridFTP is and what it is not
- Some history/background including:
    - Our motivation for developing GridFTP
    - Why we chose FTP as the basis for GridFTP
    - The various Internet RFCs on which GridFTP is based
    - A brief history of the development of GridFTP
- Finally, a peek at what we are thinking about for the future of GridFTP

Note this document *does not* discuss the protocol. For that see:

http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

GridFTP Summary

The GridFTP protocol and family of tools were born out of a realization that the Grid environment needed a fast, secure, efficient, and reliable transport mechanism. The Globus Project™ surveyed available protocols and technologies, implemented some prototypes, and settled on using FTP and its existing extensions as a base, and then extending it again to add missing required functionality. The result is a protocol and a family of tools that provide the following features:

- **Grid Security Infrastructure (GSI) and Kerberos support:** Robust and flexible authentication, integrity, and confidentiality features are critical when transferring or accessing files. GridFTP supports both GSI and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality.
- **Third-party control of data transfer:** In order to manage large data sets for large distributed communities, it is necessary to provide third-party control of transfers between storage servers. GridFTP provides this capability by adding GSSAPI security to the existing third-party transfer capability defined in the FTP standard.
- **Parallel data transfer:** On wide-area links, using multiple TCP streams can improve aggregate bandwidth over using a single TCP stream. This is required both between a single client and a single server, and between two servers. GridFTP supports parallel data transfer through FTP command extensions and data channel extensions.
- **Striped data transfer:** Partitioning data across multiple servers can further improve aggregate bandwidth. GridFTP supports striped data transfers through extensions defined in the Grid Forum draft.
- **Partial file transfer:** Many applications require the transfer of partial files. However, standard FTP requires the application to transfer the entire file, or the remainder of a file

starting at a particular offset. GridFTP introduces new FTP commands to support transfers of regions of a file.

- **Support for reliable data transfer:** Reliable transfer is important for many applications that manage data. Fault recovery methods for handling transient network failures, server outages, etc., are needed. The FTP standard includes basic features for restarting failed transfer that are not widely implemented. The GridFTP protocol exploits these features, and substantially extends them.
- **Manual control of TCP buffer size**: This is a critical parameter for achieving maximum bandwidth with TCP/IP. The protocol also has support for automatic buffer size tuning, but we have not yet implemented anything in our code. We are talking with both NCSA and LANL to see if it makes sense to integrate work they are doing in this area into our code.
- **Integrated Instrumentation**: The protocol calls for restart and performance markers to be sent back. It is not specified how often, and this is something we intend to address shortly.

What Does "GridFTP" Mean?

Unfortunately, it means a number of things and that is sometimes confusing. Here are legitimate uses of the term GridFTP:

- The *GridFTP Protocol*: This refers to the wire protocol used and is defined by a draft technical specification submitted to the Global Grid Forum. (name?)
- The *Globus Toolkit V2.0™ GridFTP Server (GT2GridFTP)*: This system is the widely used open source wuftpd FTP server code base extended to support the GridFTP protocol extensions. GT2GridFTP is distributed with the Globus Toolkit. Note that as there are many FTP server programs, similarly there can be many GridFTP server programs. Each will have its own strengths and weaknesses, and may or may not support the entire set of GridFTP protocol features. While most people have been saying "GridFTP Server" and meaning our server, it is helpful and in the case of support requests, critical, that we know exactly what server you are using. A couple of points: First, GT2GridFTP is in Beta release, and we are working towards a final release. Second, GT2GridFTP is *not* the same as gsi-wuftpd (see below).
- The *GridFTP family of tools*: We use the term "GridFTP" to refer to the entire family of GridFTP tools distributed with the Globus Toolkit: The GridFTP server, our client tools, client library, control library, etc..

The following are *not* legitimate uses (see the section on genealogy for details on where the confusion arose):

- The 1.1.3 GridFTP server: There is no such thing. This term usually refers to the gsi-wuftpd server released in May 2000 as an interim tool while we completed GridFTP. Note that this server supports GSI authentication but does *not* speak the GridFTP protocol
- The 1.1.3 GridFTP client: Again, no such thing. This term usually refers to gsi-ncftp, gsi-ncftpput, and gsi-ncftpget.

- GridFTP client: No such animal. We provide a client tool called globus-url-copy, which is part of the GridFTP family of tools, but typing GridFTP at your command prompt will not be very satisfying.

Why Did We Need a New Transport Protocol?

It is clear that applications requiring petabytes of data are emerging as an important class of applications. The Globus Project decided that the development of tools to support such applications was a good strategic fit with Globus Project goals and expertise, and began to develop a list of requirements . One of those requirements was a transport protocol that met the following criteria:

1. Targeted at bulk data transport: We saw this as a protocol to move *lots* of data (100s of Megabytes and above)
2. Based on industry standards: i.e., a clear, well defined, published, nonproprietary protocol.
3. Secure: Allowed for authentication, authorization, integrity, and privacy
4. Fast and Efficient: This meant employing multiple levels of parallelism and minimizing overhead.
5. Robust: The protocol must be able to tolerate system failures gracefully.
6. Allowed 3rd party transfers: We believe much of the traffic will be generated by automated systems such as schedulers.
7. Integrated instrumentation: The protocol must provide feedback on operational status so that intelligent actions can be taken during transfers.
8. Easily Extensible: Both in terms of standards body approval and technically/architecturally/coding wise.

Why GridFTP and not Grid "Choose Your Favorite Protocol"?

Based on the requirements listed above, we began to survey existing protocols and systems to see if any of them met all the requirements (not likely) and if not, which might be a good base from which to build. The following were examined:

1. FTP and its extensions
2. HTTP and its extensions (notably DAV)
3. HPSS (High Performance Storage System from IBM)
4. DPSS (Distributed Parallel Storage System from LBNL)
5. SRB (Storage Resource Broker from SDSC)

None of these candidates met all the requirements. However, it became clear early that there were basically two contenders: HTTP and FTP. The others all suffered from a failure to meet criteria #2, i.e., their protocol was not published and/or proprietary. We considered this to be a *major* problem, as it would make getting whatever we came up with standardized far more difficult. The decision between HTTP and FTP was not trivial, but in the end it became clear that FTP was the better starting point for us. A brief synopsis of the pros and cons is as follows:

1. Bulk Transport: FTP, yes. HTTP, no. This is related to performance, i.e., we wanted performance on large data transfers, not lots of small exchanges.
2. Standard: Yes for both, both also had been extended, with FTP having an edge.
3. Secure: Both had security. Both could have GSI integrated. This was a wash.
4. Fast: Both FTP and HTTP can do parallelism by executing multiple retrievals at once. However, that does not help when trying to move a Terabyte sized file. Neither supported striping (files split across multiple hosts). The separation of control and data channels in FTP makes 3rd party transfers trivial and allows for a single control channel with multiple data channels.
5. Robust: Both have ways of dealing with restart, but neither was considered adequate.
6. 3rd party: FTP, yes. HTTP, no.
7. Instrumentation: Neither had it, but it would be a minor addition in either case. This one was a wash.

In the end, either HTTP or FTP could have been made to work, but two items gave FTP the edge. First and foremost was the separation of the control and data channels. This enables 3rd party support, it allows the data channel protocol to be strictly for bulk data movement, and even potentially pluggable, with control messages on a separate channel, and it allows an easier implementation of the instrumentation. Second, FTP also had a history of being extended and we felt it might be easier getting extensions to FTP accepted. Bottom line: FTP could get us where we needed to be with *much* less work.

Genealogy of the GridFTP Family of Tools

The GridFTP family of tools descends from a proud line. It draws features from the following RFCs:

- RFC 959: The original FTP RFC
- RFC 2228: Security Extensions
- RFC 2389: Feature Negotiation, and support for Command Options
- IETF Draft: Stream mode restarts, standard listings (not implemented yet)

GridFTP also has some close cousins. This is a *major* source of confusion for people and trouble for the Globus Project. At the SC '99 conference the Globus Project demonstrated a prototype of something called the GridStorage architecture. It was an interesting prototype, but its primary accomplishment was to show us that it wasn't the right answer, so we went back to the drawing board and started work on GridFTP. However, GridFTP development was going to be a significant undertaking, and the community wanted a solution… soon. It turned out that we could provide one big improvement with relatively little effort, that being the addition of security. The Grid Security Infrastructure was relatively stable and RFC 2228 already specified how to add security to vanilla FTP. So, we took wuftpd and added GSI security to it, producing what we call gsi-wuftpd. We also modified the ncftp client to use GSI security and called it gsi-ncftp. Interestingly, we never made a conscious decision to support ncftpput and ncftpget (probably a mistake on our part), which are the scriptable versions, but they have kind of snuck along for the ride.

Because gsi-wuftpd was available at the same time as the Globus Toolkit version 1.1.3, many people refer to this code as "the GridFTP in 1.1.3." This is *wrong* and here is why: First, gsi-wuftpd does *not* speak the GridFTP protocol: it only supports GSI but not other minimum requirements for the GridFTP protocol draft specification. Second, gsi-wuftpd was not released as part of the Globus Toolkit V1.1.3. It was always a separate download from the web page. It did require that you have GSI installed, and most people accomplished this by installing the toolkit first, but it was *not* part of 1.1.3. The timeline at the end of the document shows this pictorially and should make this clearer.

So, in summary, here is the proper terminology:

- **gsi-wuftpd**: The wuftpd server that was patched to provide GSI security, and available in the same time frame as the Globus Toolkit v 1.1.3. It does *not* speak the GridFTP protocol, it is *not* GridFTP, and it is *not* part of any version of the Globus Toolkit.
- **gsi-ncftp**, **gsi-ncftpput**, **gsi-ncftpget**: The client tools provided with gsi-wuftpd. They were modified to use gsi security, but again, they do *not* speak the GridFTP protocol, they are *not* GridFTP, and they are *not* part of any version of the Globus Toolkit.
- **GT2GridFTP server**: The server with support for the GridFTP protocol extensions added, made available during our Globus Toolkit V2.0 alpha and beta programs and to be available once V2.0 makes final release. This server, like gsi-wuftpd**,** is based on wuftpd (the source of a lot of confusion), but note that we might produce other GridFTP servers not based on wuftpd in the future.
- **The GridFTP Protocol V1.0**: This refers to the wire protocol used and is defined by the Global Grid Forum draft document.
- **The GridFTP family of tools**: We use the term "GridFTP" to refer to our entire family of tools: The GridFTP server, our client tools, client library, control library, etc.

GridFTP Futures

The Globus Project team at ANL will cease support for the gsi-ftp tools (the interim tools that have security only, see description above) , as they represent an unacceptable support burden and cause confusion. Precisely when support will be discontinued will depend on input from the user community. It will not be until a final release of Globus Toolkit 2.0, and will *probably* be sometime in the next year.

There will be further additions to the protocol, but there are no planned changes to the existing protocol for GridFTP v 1.0. With the additions, we will have a protocol which addresses 90+% of the requirements. Our intent is that this protocol will become standard within the Global Grid Forum, and we expect it to be the workhorse for data transport on the Grid for 2-3 years, minimum. However, it will still have one major problem, which is that it the data channel connection and data flow must be in the same direction. In other words, the data channel is not bi-directional. This can cause problems in firewall situations. There are also additional features and developing technologies that would be useful, such as a full duplex protocol, pipelining of commands, web services, etc..

In order to resolve these issues we will be developing additional protocols. The GridFTP protocol is actually a protocol consisting of multiple underlying "sub-protocols". There is a protocol for the control channel, and multiple protocols for the data channel: stream mode, block mode (which no one, including the Globus Project implements), and Extended Block Mode (one

of our extensions). We will develop a new data channel protocol that enables bi-directional data transfer, pipelining of commands, all the existing functionality of Extended Block Mode, and possibly other features as well. We will also likely be providing a web service interface to GridFTP (SOAP/XML RPC's). How exactly we go about doing this will depend on the needs of the community when the new protocols are ready. If there is a need for backward compatibility we may integrate the new protocol as a new mode into the existing GridFTP protocol, with automatic protocol negotiation. We could then provide a web services front end that executes our existing API (this is what our Reliable File Transfer prototype does). We could also implement a SOAP/XML based control channel directly in the server, with or without a "standard" control channel as an alternative. Protocol negotiation would again be used to determine which protocol would be used, if both were available/needed.

The Globus Project is fully cognizant of the fact that people will be investing significant time and energy into developing code based on our GridFTP protocol and our API's, which implement that protocol. We also appreciate the value of standardized protocols. However, the protocols cannot remain stagnant and must evolve to meet future needs. We are committed to meeting these future needs and evolving these protocols, while providing reasonable migration paths and minimizing the impact on the existing code base.

The following Gantt chart summarizes this and the discussion of genealogy and futures. The timelines do not represent commitments on our part!