GFD-I.083
Firewall Issues FI – RG

Ralph Niederberger (Editor), Research Center Jülich
William Allcock, Argonne National Laboratory
Leon Gommans, University of Amsterdam
Egon Grünter, Research Center Jülich
Thijs Metsch, German Aerospace Centre – DLR e.V.
Inder Monga, Nortel Networks
Gian Luca Volpato, Christian Grimm, RRZN LUH

August 16, 2006

Firewall Issues overview.

Status of This Memo

This document provides information to the Grid community on issues which grid applications may have when dealing with firewalls. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

Several kinds of network devices like Firewalls are used to protect an institutional network against malicious attacks from the public Internet. This document enumerates and illustrates a selected set of grid scenarios that encounter some issues when dealing with firewall types of devices. The knowledge and experience gathered through these use-cases is utilized to classify the issues into homogeneous categories that can be used by grid application developers and management personnel as guidance. These categories will be used to propose new or recommend existing academic and/or standards based solutions to the grid community.

Contents

1 Introduction

Grid-Projects with external partners lead to communication relationships between external and internal computer systems often requiring special configurations at firewall systems. These configurations include:

- allowing access for communication sessions (ports)
- allowing access to single systems or whole sub networks

Additionally physical access may be provided by implementing dedicated high-performance physical or logical links as fiber, wavelength, sub wavelength, VPN, VLAN, etc. Assuming that external sources cannot gain access and misuse these links they are rarely secured by firewalls.

Because of the limitations of today's firewalls (often limited to 1 Gb/s throughput, some products already offering 10 Gb/s) load balancing by means of multiple firewalls is often based on IP or MAC-address balancing, i.e. one stream will be executed by one firewall giving real balancing only with multiple communication streams. Grid applications with huge bandwidth demands (one single data stream) do not benefit from these types of firewalls. Some firewall clusters implement round-robin mechanisms, but they are limited to lower throughput because of the extreme overhead needed for status information updates between the different firewall components.

Only few firewall systems are able to handle applications with dynamically assigned ports. Some implementations exist for applications such as FTP, H.323, and SIP. But currently no general solution is available; support for protocols used by Grid applications may not be expected in near future either.

Often within a grid environment each institution or even worse each installation has its own firewall system. All of them have to be traversed by Grid applications. Because of the problems discussed above, project networks are placed in a demilitarized zone in most of the cases. This implies that every computer system used in the project has to be secured carefully. Wrongly configured systems lead to immediate security vulnerabilities.

Supercomputers or special systems may be connected via dedicated networks assuming a "Net of Trust", i.e. users at these systems will be trusted leading to insider security problem. Compromise of these systems leads to increased security problems.

Finally the situation shown above result in:

- administrative overhead for deployment and protection of grid environments
- wildcard access rights (ports not known, so access granted to whole system)
- weaker policies or no security policies anymore
- general decreasing security level to that of the partner installation
- security vulnerability because of open ports for long time periods.

The examples above show there are new demands on today's firewalls. Many national and international activities/projects try to cope with these problems. Some of them are:

- D-Grid, a German project funded by BMBF, Germany, work package FG3-5, "Design and deployment of firewall concepts within grid environments, Performance and dynamic configuration" (see: http://www.d-grid.de)
- MIDCOM (see: IETF Internet draft "Middlebox Communication: Framework and Requirements" at http://sip-router.org/info/players/ietf/firewall/midcom/draft-kuthan-fcp-02.txt)
- University of Buffalo, Grid Computing Research projects, ACDC-Grid Firewall - „Advanced Computational Data Center Dynamic Firewall (ACDC Dyna-Fire) Development" (see: https://grid.ccr.buffalo.edu/research/)

This document tries to identify typical scenarios of today's grid environments. It structures these scenarios into use cases and classifies these cases into general communication concepts used by grid applications. These classifications provide a fundament for further investigation into possible solutions that will be discussed within a later FI-RG document.

The solutions examined so far can be divided into three categories:

1. Solutions that do not require any modification or additional software/hardware development (e.g. give access to a special port)
2. Solutions through development of new software/hardware components, which allow handling of special use cases or classes of use cases (e.g. as it's done for the ftp protocol by checking the control channel and opening the ports negotiated between the communication partners via the control connection).
3. No solution is achievable with the current types of firewalls. New software/hardware models have to be developed.

The current document tries to pave the way to these classifications, to identify which of the current grid applications fall into which category and how to deal with use cases which are categorized into categories 2 and 3 above.

2 Definitions

The goal of this chapter is to provide an overview of the types of devices and software components that are used to protect grid applications and infrastructures from malicious attacks from the Internet.

2.1 Firewall

A **firewall** is a logical object (hardware and/or software) within a network infrastructure which prevents communications forbidden by the security policy of an organization from taking place, analogous to the function of firewalls in building construction. Often a firewall is also referred to as a **packet filter**.

The basic task of a firewall is to control traffic between different zones of trust and/or administrative authorities. Typical zones of trust include the Internet (a zone with no trust) and an internal network (a zone with high trust). The ultimate goal is to provide controlled connectivity between zones of differing trust levels through the enforcement of a security policy and a connectivity model based on the least privilege principle.

Proper configuration of firewalls demands skill from the administrator. It requires considerable understanding of network protocols and of computer security. Small mistakes can lead to a firewall configuration worthless as a security tool and, in extreme situations, fake security where no security at all is left.

2.1.1 Classification of firewalls

There are three basic criteria to categorize firewalls:

1. whether the communication occurs between a single host and a network, or between two or more networks;
2. whether the communication is intercepted at the network layer or at the application layer;
3. whether the communication status is tracked at the firewall or not.

With regard to the position of a firewall in the network layout there are:

- host firewalls, i.e. software applications which filter traffic entering or leaving a single computer;
- network firewalls, i.e. software normally running on a dedicated network device or computer positioned on the boundary between two or more networks or DMZs (demilitarized zones). Such a firewall filters all traffic entering or leaving the connected networks.

The latter definition corresponds to the conventional, traditional meaning of "firewall" in networking. Additionally, firewalls may be located between administrative domains of an organization (e.g. between production, research, administration and finance departments).

In reference to the software layer where the traffic is intercepted, three main types of firewalls exist:

- network layer firewalls
- application layer firewalls
- application firewalls

The network-layer and application-layer types of firewalls may partially overlap. Indeed there are examples of single systems that implement both of them together.

Application firewalls are sometimes used in wide area networks (WAN) to govern the access to the system software. An extended description would place them at a lower level than application-layer firewalls, actually at the operating system layer, and thus they could otherwise be called operating system firewalls.

Lastly, depending on whether the firewalls track communication status, two categories of firewalls exist:

- stateful firewalls
- stateless firewalls

2.1.1.1 Network layer firewalls

Network layer firewalls operate at a (relatively low) level of the TCP/IP protocol stack as IP-packet filters, denying packets to pass through the firewall unless they match one positive filtering rule. The firewall administrator defines the rules or default built-in rules are applied (as in some inflexible firewall systems). A more permissive setup could allow packets to pass the filter as long as they do not match one "negative rule" or "deny rule".

Today network firewalls are included into most computer operating systems and network appliances.

2.1.1.2 Application-layer firewalls

Application-layer firewalls work at the application level of the TCP/IP stack and intercept all packets traveling to or from an application (HTTP traffic, telnet traffic, ftp traffic, etc.). They block packets, which do not conform to the application’s network protocol, usually dropping them without acknowledgement to the sender. In principle, application-layer firewalls can stop all unwanted incoming traffic from reaching protected machines.

By inspecting all packets for improper content, these firewalls can even prevent the spread of viruses. In practice, however, this task becomes so complex and so difficult to attempt (given the variety of applications and the diversity of content each of them may allow in its packet traffic) that comprehensive firewall design does not generally attempt this approach.

The XML Firewall exemplifies a recent kind of application-layer firewall.

2.1.1.3 Application firewalls

The term application firewalls is often used to describe security tools that control access to special services that run on an operating system. They are composed of software components securing the local system by checking which external (remote) hosts may access the services running on this node. Often these firewalls are called operating system firewalls.

A well-known implementation of application firewalls is TCP wrapper.

2.1.1.4 Stateful/stateless firewalls

Modern network-layer firewalls can filter traffic based on many packet attributes like source IP, source port, destination IP or port, destination service like WWW or FTP. They can filter based on protocols, TTL values, netblock of originator, domain name of the source, and many other attributes.

Having the ability to look into the packets in more detail, allows monitoring the status of the transmission (based on TCP options or simulated status for stateless protocols) and implementing more complex filtering rules. A stateful firewall usually allows incoming TCP packets only when they belong to a connection started by a host in the protected network. Connection requests coming from untrusted networks are rejected.

In contrast to this behavior a stateless firewall does not monitor the status of connections. Every packet has to be checked and mapped to a rule that either allows or denies it.

2.2 Firewall (global definition)

In a broader sense a firewall is the implementation of an institution's security policy concerning traffic exchange between different security domains. It is not only a black box or a single hardware component. It can be much more. It is the set of all rules to be enforced for secure communication. It is the way to check the compliance with these rules and it is the whole collection of software and hardware used to implement this mission.

2.3 Network Address translators

Network Address Translation (NAT) [RFC 1631] provides a way to map IP addresses from one IP network to another IP network allowing transparent routing between client and server hosts in distinct networks. Transparent in this context describes the possibility of mapping IP addresses without explicit interaction or knowledge of the user. In this sense NAT is transparent to the user. In the simplest case, a user behind a NAT initiating a single simple message, this translation within the IP header does not lead to any problems within the transaction. But often IP addresses are overloaded with security issues, e.g. access rights, user authentication and last but not least authorization. In this scenario NAT may become a problem, especially as often configured, when mapping is done dynamically. NAT in general is not a security mechanism, but nevertheless introduced additional barriers against intruders. It hides the internal network to external hackers.

NAT is often applied to connect private networks, using private address spaces, to the external Internet with officially registered addresses. Using this technique it is possible to temporarily solve the current shortage of official IP addresses by recycling them for different hosts until new IP addresses (IPv6) become available and commonly deployed. Address translation is normally done at the borders of private domains; this particular modus operandi makes this technique appealing also for security reason. The main advantage of NAT is that it can be enabled without any changes to routers or hosts. Unfortunately NAT cannot be used in conjunction with all existing applications since some services encode IP addresses in the packet payload. In these cases NAT must co-exist with application level gateways (ALGs, see below).

Allowing transparent routing, NAT devices modify host addresses in the packets on the fly and maintain state information of communication flows. Packets belonging to the same communication stream have to be translated in the same manner, i.e. to the same IP address.

Port Address Translation (PAT) or Network Address Port Translation (NAPT) [RFC 2663] enhances this technique. Here different hosts are mapped to the same IP address, using port information (source and destination port) to differentiate between different streams of communication (e.g. TCP and UDP port numbers, ICMP query identifiers). Many internal private IP addresses can be translated to one single official external IP address.

Often NAT and PAT are used as a security mechanism. Internal hosts are allowed to setup communication paths to external hosts, but connections from external hosts to internal hosts can be setup only if a translation is currently active (i.e. an internal host has already setup a connection). Here NAT is done dynamically, making it harder for an attacker to target any specific host in the NAT domain. NAT routers may be used in conjunction with firewalls to filter unwanted traffic. Often the firewall itself offers the NAT functionality.

Problems arise with end-to-end IPsec, because this protocol does not allow the presence of NAT devices in the communication path. IPsec encodes the source and destination addresses of the end-to-end communication. If NAT changes one of these addresses the IPsec module will detect it and the communication will fail. A possible solution is to use NAT devices as end-points of the IPsec tunnel.

“NAT devices, when combined with ALGs, can ensure that the datagrams injected into Internet have no private addresses in headers or payload. Applications that do not meet these requirements may be dropped using firewall filters. For this reason, it is not uncommon to find NAT, ALG and firewall functions co-exist to provide security at the borders of a private network. NAT gateways can be used as tunnel end points to provide secure VPN transport of packet data across an external network domain. (RFC 2663)”

2.4 Application level gateways

Not all applications lend themselves easily to translation by NAT devices, especially those applications that encode IP addresses and port numbers in the payload. In these cases, Application Level Gateways (ALGs) play a very important role. ALGs are application-specific translation agents that transparently allow an application running on a host in one address realm to connect to its counterpart running on a host in a different realm. ALGs may interact with NAT to set up state, use NAT state information, modify application specific payload and perform whatever else is necessary to get the application running across disparate address realms.

ALGs may not always utilize NAT state information. They may glean application payloads and simply notify NAT to add additional state information in some cases. ALGs are similar to proxies, in that both ALGs and proxies facilitate application specific communication between clients and servers. Proxies use a special protocol to communicate with proxy clients and relay client data to servers and vice versa. Unlike proxies, ALGs do not use a special protocol to communicate with application clients and do not require changes to application clients.

In the context of firewalls application level gateways are used to open up pinholes into the firewall. Both kinds of usage introduce some major problems into the organizations security policy. First of all by rewriting payload, the message content will be changed which breaks message authentication. The receiver cannot check anymore if the real message (data) part has been modified by a third party. Additionally they force firewall vendors to support a bunch of application protocol stacks on their firewalls, which is expensive, time consuming (processing and throughput) and increases exposure to implementation errors.

Nevertheless there are grid scenarios, in which ALG cannot be avoided to establish the required communication paths of particularly projects.

2.5 VPN gateways

A Virtual Private Network (VPN) gateway can be considered as the "employee entrance" into a corporate network, whereas a firewall could be considered as the "public entrance". A corporate network is typically classified as a private network, created to support the business of an Enterprise, SMB, or any other organization with a need to protect its networked resources from public access. A VPN gateway uses credentials issued by the Corporate Network Administrator to create a security association between the corporate VPN gateway and a remote VPN site. Remote sites can either be individual PC clients or other VPN gateways. A remote VPN gateway allows the corporate network to be securely extended into a branch office via an insecure network. This setup is called a site-to-site VPN. A PC VPN client allows an individual employee to access the corporate network from the Internet when at home or traveling. Protocols, such as IPsec, L2TP, PP2P and SOCKS ensure authenticated and encrypted communication between VPN sites by creating a tunnel. On such connections, packets are constructed in a specific VPN protocol format and are encapsulated within some other base or carrier protocol, then transmitted between VPN client and server, and finally de-encapsulated on the receiving side. The base protocol for Internet is IP. Other cases, which typically rely on point-to-point connections, may use layer 2 protocols.

Most VPN gateways offer functionalities similar to firewalls; packet filtering and packet inspection are examples. It is therefore important to consider these types of devices within the scope of this document.

3 Grid applications and their issues with firewalls

This chapter contains input from various organizations describing their issues with firewalls. This input may describe problems and suggested solutions. When contributions were asked, no structure was suggested as to keep the input as broad as possible. The information within this section will be analyzed and classified in subsequent sections.

3.1 Middlewares and Protocols

3.1.1 The Globus Toolkit

Organization: Research Center Jülich GmbH, Jülich, Germany

The Globus Toolkit homepage (see: http://www.globus.org/toolkit/) welcomes the user with the following introduction:

> "The Globus Toolkit® is an open source software toolkit used for building grids. It is being developed by the Globus Alliance and many others all over the world. A growing number of projects and companies are using the Globus Toolkit to unlock the potential of grids for their cause."

Designed as a toolkit and developed by many participants it has gone through various steps of programming versions, which led to different security solutions throughout the design process. The toolkit allows users to start compute jobs, request status information and cancel jobs, transfer data between grid resources and manipulate their availability and access criteria.

Because of its widespread deployment it is extremely worthy to examine which protocols have to be enabled and/or which ports have to be opened at the local firewalls, to allow the Globus Toolkit to be used at local sites. A very detailed description concerning firewall requirements for the different Globus Toolkit components has been given by Von Welch in the "Globus Toolkit Firewall Requirements" document

(see: http://www.globus.org/toolkit/security/firewalls/Globus%20Firewall%20Requirements-7.pdf).

Globus differentiates between Grid Service ports, Ephemeral ports and Controllable Ephemeral ports. A grid service port is a static, single, port defined for a very specific grid service (comparable with ftp, ssh or http ports). An ephemeral port is a port that will be dynamically assigned by the system during program execution. A controllable ephemeral port is a special case of ephemeral ports that can be configured to be within a predefined port range.

Depending on the Globus application used, different communication scenarios can be identified. These different kinds of communication streams vary in the complexity of the firewall configurations required at the server and client sides. Communications can be initiated:

1. from an ephemeral port on the client to a grid service port on the server;
2. from a controllable ephemeral port on the client to a grid service port on the server;
3. from a controllable ephemeral port on client to a controllable ephemeral port on the server.

The first group corresponds to connection to standard services. The server firewall must allow connections from remote arbitrary ports to local static ports. The client firewall must allow connections from local arbitrary ports to remote static ports.

The second group corresponds also to standard service communication, the only difference being the arbitrary port used by the client (controllable ephemeral port) lies within a small subrange of ports. The server firewall must allow connections from remote arbitrary ports to local static ports. The client firewall must allow connections from local range of arbitrary ports to remote static ports.

The third group is the most complex one. First of all, any installation may define its own port ranges, with ranges differing from server to server (when several servers are in place). As a result the complete list of access rules for the firewall becomes quite complicated. The server firewall must allow connections from remote arbitrary ports to local ranges of arbitrary ports. The client firewall must allow connections from local range of arbitrary ports to remote arbitrary ports. For some services the magnitude of the port range depends on the number of parallel sessions initiated. Additionally these port ranges are opened outside of the well-known port range (0-1024), allowing unprivileged applications to use these ports too.

From the use cases point of view the Globus Toolkit behaves as most other applications discussed so far. Some Globus applications can be handled securely in a standard way of firewall configuration. Others need techniques which, implemented securely and automatically manageable, are not available yet.

3.1.2 **UNICORE**

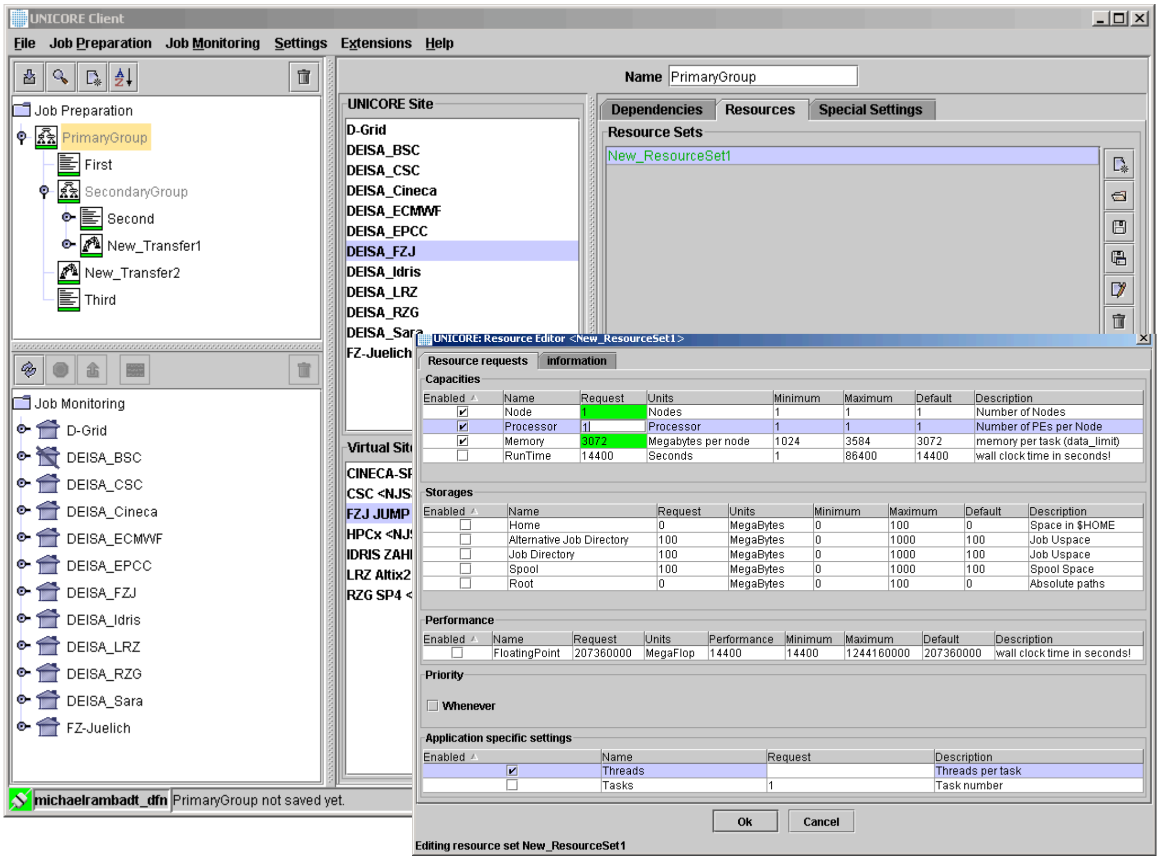
Organization: Forschungszentrum Jülich GmbH, Jülich, Germany

The UNICORE software (UNiform Interface to COmputing Resources, is a software interface which allows easy and uniform access to distributed computing resources, and which provides support for running scientific and engineering applications in a Grid environment (see also: UNICORE – The seamless Grid solution, http://unicore.org). Scientists can use different supercomputers as well as other computing and storage resources without having to become experts in the special kind of access software and security policies of the various (super-) computer centers.

UNICORE provides a science and engineering Grid combining resources of supercomputer centers. It makes these resources available through the Internet. UNICORE uses a strong authentication and authorization scheme in a consistent and transparent manner. Differences between platforms are hidden from the user. A seamless HPC portal for accessing supercomputers, compiling and running applications, and transferring input/output data has been developed.

Through using UNICORE end-users can concentrate onto their real application issues and therefore increase their productivity. Internal supercomputer specifics are hidden to these end-users who don’t need to learn any kind of job control languages.

The UNICORE user prepares or modifies structured jobs through a graphical client interface on his local workstation or PC. Besides the UNICORE internal job description UNICORE also is able to handle XML-Jobs. After preparation the created job has to be submitted to one of the platforms of a UNICORE Grid. Here the user may monitor and control the submitted jobs through a second area in the UNICORE client.



Name PrimaryGroup

michaelrambadt_dfn PrimaryGroup not saved yet.

Capacities

Enabled	Name	Request	Units	Minimum	Maximum	Default	Description
☑	Node	1	Nodes	1	1	1	Number of Nodes
☑	Processor	1	Processor	1	1	1	Number of PEs per Node
☑	Memory	3072	Megabytes per node	1024	3584	3072	memory per task (data_limit)
□	RunTime	14400	Seconds	1	86400	14400	wall clock time in seconds!

Storages

Enabled	Name	Request	Units	Minimum	Maximum	Default	Description
□	Home	0	MegaBytes	0	100	0	Space in $HOME
□	Alternative Job Directory	100	MegaBytes	0	1000	100	Job Uspace
□	Job Directory	100	MegaBytes	0	1000	100	Job Uspace
□	Spool	100	MegaBytes	0	1000	100	Spool Space
□	Root	0	MegaBytes	0	100	0	Absolute paths

Performance

Enabled	Name	Request	Units	Performance	Minimum	Maximum	Default	Description
□	FloatingPoint	207360000	MegaFlop	14400	14400	1244160000	207360000	wall clock time in seconds!

Priority

□ Whenever

Application specific settings

Enabled	Name	Request	Description
☑	Threads		Threads per task
□	Tasks	1	Task number

Editing resource set New_ResourceSet1

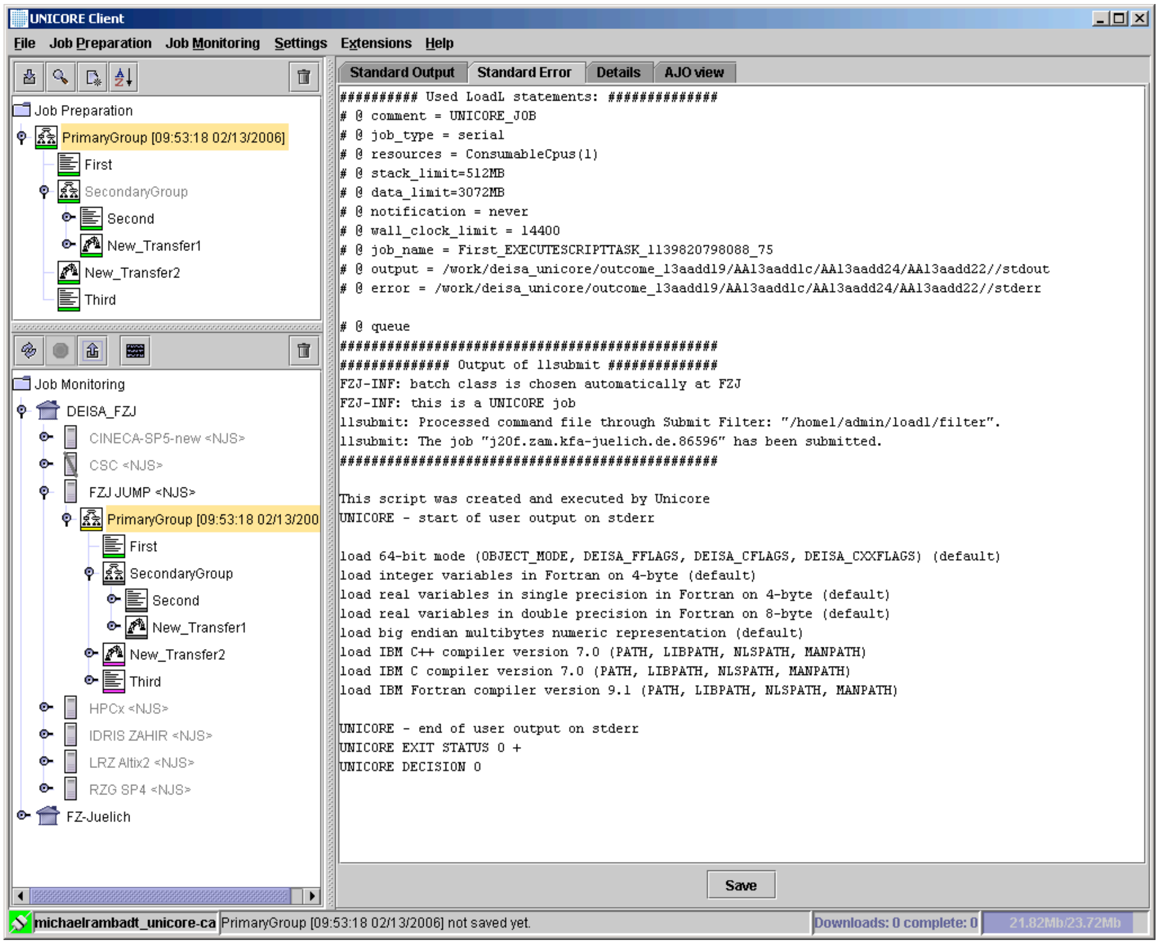
Figure 7: Job preparation: Definition of a job, adding dependencies and resource requests.

UNICORE allows to structure jobs, dividing them into independent tasks. Dependencies between these tasks can be assigned. The structured model allows executing a job, divided into subtasks to be run on different locations of the UNICORE Grid leading to hierarchical job structures and data locations. So UNICORE is able to manage complex multi-site and multi-step workflows efficiently.

UNICORE has a three tier architecture which consists of the Client, the Gateway, and the NJS /TSI. The NJS (Network Job Supervisor) is responsible for mapping the abstract job description to concrete target system issues. This is done with the IDB (Incarnation Database). For mapping, it also authorizes the user to access the target system. The NJS is the front-end to the target system. The TSI (Target System Interface) is a library of Perl modules installed on the target system (e.g. the supercomputer) itself and providing an interface between the batch system and the UNICORE servers. Because all UNICORE components, except the TSI, are implemented in Java, the UNICORE Client and servers are very platform independent.

UNICORE tasks and resources are represented in abstract terms and units, so that a server can translate them into the platform-specific commands and options. Input and output files are automatically imported/exported from/to the user's file space or transferred from earlier tasks of the same job. Explicit transfer tasks handle the high-speed data transfer between different sites. The UNICORE servers select the most efficient mechanism for each transfer.

For each job, the user specifies the intended target system and the task's resource requirements. The client software checks whether the resources requests by the end-user can be satisfied by the target system, and submits the job into the target system. To resubmit a job at a different system, the user simply changes the target system.



```
########## Used LoadL statements: ##############
# @ comment = UNICORE_JOB
# @ job_type = serial
# @ resources = ConsumableCpus(1)
# @ stack_limit=512MB
# @ data_limit=3072MB
# @ notification = never
# @ wall_clock_limit = 14400
# @ job_name = First_EXECUTESCRIPTTASK_1139820798088_75
# @ output = /work/deisa_unicore/outcome_13aadd19/AA13aadd1c/AA13aadd24/AA13aadd22//stdout
# @ error = /work/deisa_unicore/outcome_13aadd19/AA13aadd1c/AA13aadd24/AA13aadd22//stderr

# @ queue
##################################################
############## Output of llsubmit ##############
FZJ-INF: batch class is chosen automatically at FZJ
FZJ-INF: this is a UNICORE job
llsubmit: Processed command file through Submit Filter: "/homel/admin/loadl/filter".
llsubmit: The job "j20f.zam.kfa-juelich.de.86596" has been submitted.
##################################################

This script was created and executed by Unicore
UNICORE - start of user output on stderr

load 64-bit mode (OBJECT_MODE, DEISA_FFLAGS, DEISA_CFLAGS, DEISA_CXXFLAGS) (default)
load integer variables in Fortran on 4-byte (default)
load real variables in single precision in Fortran on 4-byte (default)
load real variables in double precision in Fortran on 8-byte (default)
load big endian multibytes numeric representation (default)
load IBM C++ compiler version 7.0 (PATH, LIBPATH, NLSPATH, MANPATH)
load IBM C compiler version 7.0 (PATH, LIBPATH, NLSPATH, MANPATH)
load IBM Fortran compiler version 9.1 (PATH, LIBPATH, NLSPATH, MANPATH)

UNICORE - end of user output on stderr
UNICORE EXIT STATUS 0 +
UNICORE DECISION 0
```

Figure 8: Job monitoring: inspect the status of running jobs and retrieve the output

In any case, the users can monitor and control their jobs through the job monitor interface, which depicts the job status graphically.. After job execution the output data of the job can be retrieved to the local workstation

User authentication is performed using X.509 certificates. Each UNICORE user has a personal user certificate signed by a trusted CA. The administrator himself is responsible to define the “Trusted CAs” in the UNICORE servers. Each job the user sends into the UNICORE Grid is signed by the private key of the certificate. User authorization is handled by the participating sites using their proven mechanisms. In this case UNICORE also completely retains the sites autonomy with authorizing users and allocating resources to them. The UNICORE interface for the user authorization is called UUDB (UNICORE User Database). This component maps the user’s public key of his personal certificate to the real Xlogin on the target system. So every time a job arrives in the UNICORE Grid the certificate is checked and compared with the entry in the UUDB. Only if both are identical will the job be transferred to the target system. To transfer jobs, control information and application data, SSL is used to guarantee data integrity and confidentiality. The signing of job representations with the originating user's private key also prevents third parties from tampering with the job contents.

The UNICORE gateway component authenticates connection requests by checking if the incoming certificate is signed by a trusted CA. The Gateway also checks that the presented user's certificate has not been revoked and is still valid. The gateway can cooperate with firewalls to permit only legitimate UNICORE traffic. It may reside outside the protected zone, in a demilitarized zone, or within the protected zone depending on the site's security setup. Using UNICORE only one port for the gateway has to be opened in the firewall.

While the Client-Gateway connection must be SSL-secured, the connection between Gateway and NJS can be optionally SSL-secured. The UNICORE NJS is generally located in the safe intranet; nevertheless it might be necessary or wished by the site's administrators to secure the Gateway-NJS connection via SSL as well. This is also one example of how UNICORE does not influence the sites autonomy. Since both the Gateway and the NJS component are provided with a server certificate, the SSL handshake can be established between those components, too.

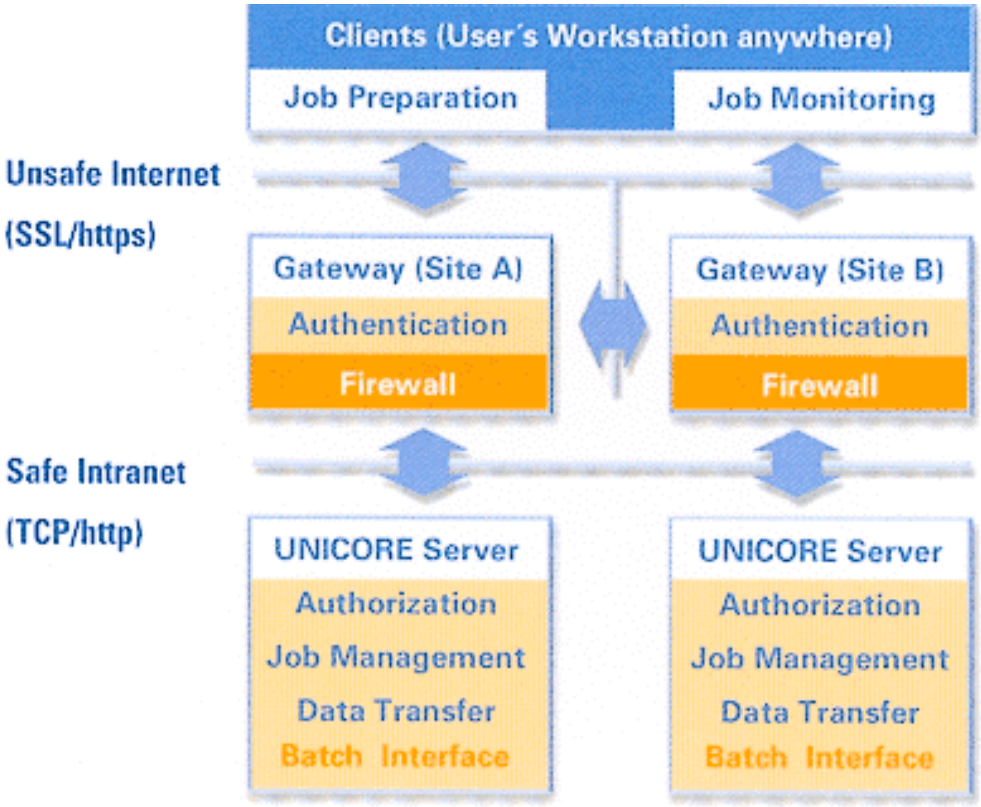


Figure 9: UNICORE architecture: system components and their interaction

The UNICORE client enables the user to create, submit and control jobs from any workstation or PC on the Internet. Only an installed UNICORE client is required. All user certificates are stored in the UNICORE client keystore. So the user might just export this keystore to e.g. a memory stick and import it on another machine and he is able to access all his jobs and resources again.

The client connects to a UNICORE gateway, which authenticates both users and other UNICORE servers, before contacting the UNICORE NJS, which in turn manages the submitted UNICORE jobs. NJS incarnates abstract tasks destined for local hosts in batch jobs and run them on the native batch subsystem. Tasks to be run at a remote site are transferred to a peer UNICORE gateway. All necessary data transfers and synchronizations are performed by the servers. They also retain status information and job output, passing them to the client upon user request.

The protocol between the components is defined in terms of Java objects. A low-level layer called the UNICORE Protocol Layer (UPL) handles authentication, SSL communication and transfer of data as inlined byte-streams and a high-level layer (the Abstract Job Object or AJO class library) contains the classes to define UNICORE jobs, tasks and resource requests.

Third-party components can be integrated into the system: on top of UPL to create alternatives to the AJO layer, or within the AJO layer defining new classes. Thus, the functionality of clients and servers can be extended within the UNICORE framework by implementing so called Plug-ins. Plug-ins are also Java objects which allow integrating different applications into the UNICORE Grid software easily.

3.1.3 **Web services Firewall Issues**

Organization: Argonne National Laboratory, ANL, US

As the web service protocol will most probably be used for the control channels and control-planes that manage GridFTP endpoints and/or dynamic firewall configurations, it is important to understand the issues and the associated requirements.

3.1.3.1 Internal vs. External EPRs

The application service’s EPR (End Point Reference) has an address that is used as the network endpoint for that service by the clients. As a result, when a service is located behind a firewall, external clients (i.e. outside the corporate firewall) cannot use the same EPR that is used by the internal clients. If the access by external clients is allowed through an application-layer firewall, then the external clients will have to be supplied with an external-EPR for the application service that will direct the clients to send their messages to the SOAP-Proxy service that, after policy enforcement, will forward the requests to the application service behind the firewall.

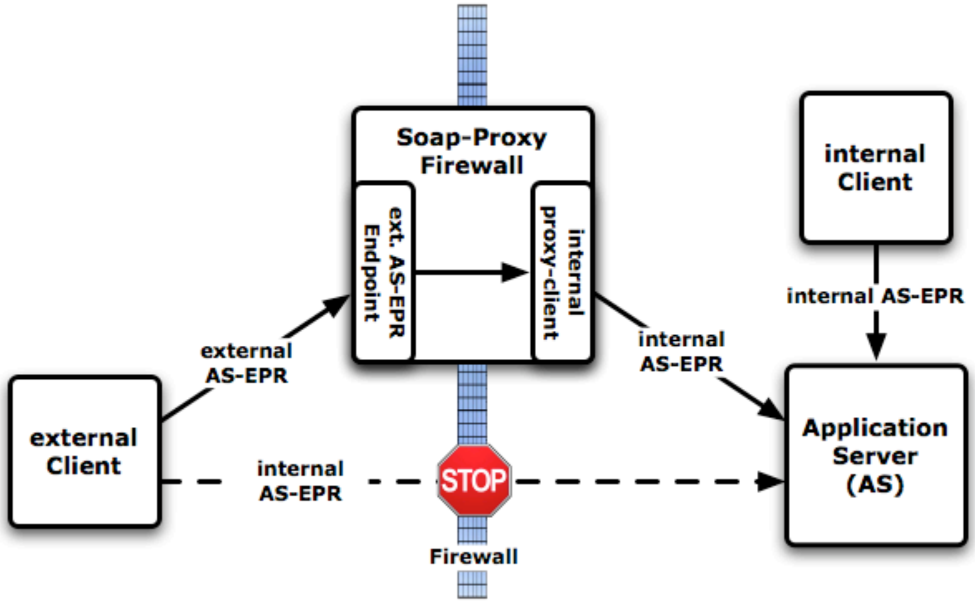


Figure 10: External Clients and Internal EPRs

The issue is depicted in Figure 10, where the external client's use of the application service's internal-EPR is blocked by the firewall, while the external-EPR is shown to route the external client's messages through the proxy service to the application service.

We have no standardized ways yet to:

- Augment the EPR with routing information
- Obtain an external EPR from an internal one
- Publish and discover the need for external EPRs
- Express policies to tell clients to extend the security context end-to-end

3.1.3.2 Ephemeral Internal EPRs

Even if we have a way to tell the external client that a soap-proxy service should be used to connect to the internal application service, we have an additional issue with factory-like patterns.

In a factory-pattern, a service is used to obtain a new EPR for a newly created or located resource. In other words, an EPR for that new service is returned in the message exchange with the factory service.

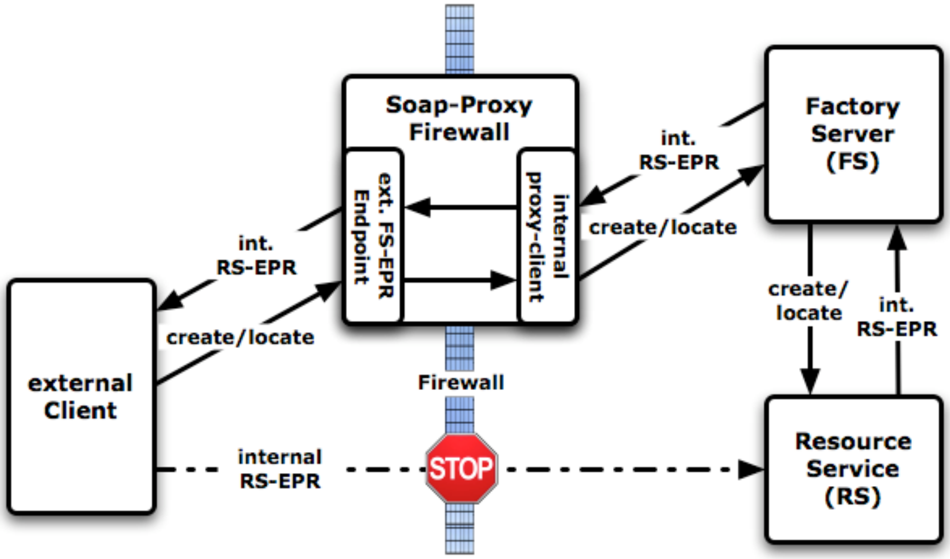


Figure 11: External Clients and Ephemeral EPRs

As shown in Figure 11, the issue with the returned EPRs is that by default they will be internal EPRs, and they should somehow be translated to external EPRs before the external client is able to use them.

The issues with ephemeral EPRs are:

- We have no standardized way for the firewall to discover which internal EPRs should be translated on the fly into external ones (feels like HTML rewriting for reverse-webproxies…)
- We have no way to express and enforce a policy that allows firewalls to deal with ephemeral EPRs, which may refer to a resource that is not in the same hosting environment as the factory, or not even on the same host, and may not even use the same identity credentials.

3.2 Data Transfer and Storage

3.2.1 GridFTP versus the Firewall

Organization: Argonne National Laboratory, ANL, US

GridFTP is a fairly troublesome application from the point of view of firewalls. It can use a significant number of ports that are in the ephemeral range and with today's protocols it is impossible neither to know in advance the full 5 tuple that describes a connection, nor to limit the usage to two ports.

GridFTP, like FTP, has two channels, a control channel and a data channel. The control channel is relatively painless. It is always a single socket connection to a well-known port. The connection is strongly authenticated, it is encrypted, integrity protected, and requires very low bandwidth, so this is something that firewall administrators are generally willing to deal with, and because of its low bandwidth, the firewall generally does not introduce any performance limitations.

The troublesome part of GridFTP is the data. Why is it so difficult? There are several reasons. First, the data channel is a logical construct and can consist of an arbitrary number of sockets, which can vary in time. The protocol allows sockets to be added or removed arbitrarily anytime during a transfer. Second, the protocol requires that the sending sides perform the TCP connect so you do not have the option of having the client be passive to work around firewall restrictions. Third, the full 5-tuple for a given connection is known very late in the process and nothing has global knowledge of the connections between individual sockets that make up the logical data channel.

Some background on how GridFTP works will help explain this. We will describe a third party transfer (a transfer between two servers mediated by the client). It is the most complex of the transfers and client/server transfers simply do only one half of the PASV/PORT command, since the client knows the other half internally when it is involved in the actual movement of data. We will describe a striped transfer, which involves *m* hosts on one end sending to *n* hosts on the other end. *m* and *n* are not required to be equal and can be one, this is a non-striped transfer for the purposes of this discussion, i.e. a striped transfer with *m* and *n* equal to one.

The client attempts to open a control channel connection on a well-known port. Assuming this port is open on the firewall and it can establish a connection, it begins sending a series of commands that do authentication, and then begin to describe the transfer, like is it binary or ASCII, etc. If this server is the receiving server, it will send the SPAS (striped passive) command. Each host at the receiving side will then listen on a set of arbitrary ephemeral ports, and the list of listening ports is sent back to the client in the response to the SPAS command. At this point that server knows it will be contacted, but it does not know by whom.

The client now attempts to open a control channel to the sending server again on a well-known port. It authenticates and begins its command sequence to the server, but this time it will send the SPOR (striped PORT) command. This command includes the list of listening ports that was returned in the response to the SPAS command. This tells the server that it *MAY* connect to this list of servers. It may connect to one, all, or some subset depending on the layout of the data. It does not yet know how many connections to make. That is determined when the OPTS RETR (retrieve options) command is sent. This indicates the minimum number of streams, the start number of streams, and the maximum number of streams. Note that it is the sending server that can decide to change the number of streams, within the limits specified, the client can not tell the server to add or remove streams; this means that there is no command sequence that can be trapped on the control channel to know when a new connection is being initiated. Once the RETR <filename> command is received, each host on the server side will determine which hosts in the SPOR list it needs to connect to and will initiate the connection, which will again be an arbitrary ephemeral port. It is only in the socket call when the connecting ephemeral port is chosen that the full 5-tuple is known.

The problem, that not only one single data port can be used, is that you can only have one process using a port. The way the control channel works is that some daemon (typically inetd) is listening on the well-known port. It gets a *single* connection, does a fork/exec, duplicates the socket, hands it off to a new process and then closes it’s file descriptor. It is now ready to accept another connection on that port from anywhere other than a host and port that already has a connection to its port 2811.

However, let’s assume that we wanted to have 2812 be the data channel port. The process listening on that port would need to be able to accept a connection, know which transfer that connection is associated with, and how many total connections were expected (all connections would have to be formed up front, this would not allow for additional connections later, a limitation of what the protocol allows, though probably not a big one). Once it had all the connections for a given transfer, it could then fork/exec a data node (GridFTP backend) dup all the necessary sockets to it, then it closes its socket, and that backend could go merrily upon its way. The problem is that there is no way, today, to know what transfer a connection is associated with and no way for that listener to know how many connections it should get.

3.2.2 Impact of dCache deployment

Organization: Forschungszentrum Jülich GmbH, Jülich, Germany

dCache is a joint venture between the Deutsches Elektronen-Synchrotron (DESY) and the Fermi National Accelerator Laboratory (Fermilab). dCache has been selected as a data storage solution to be used in the German D-Grid project started in 2005.

dCache allows storing and retrieving huge amounts of data, distributed among a number of heterogeneous server systems. These systems simulate a single virtual file system. Depending on the Persistency Model, dCache provides methods for exchanging data with backend (tertiary) Storage Systems as well as space management, pool attraction, dataset replication, hot spot determination and recovery from disk or node failures. Connected to a tertiary storage system, the cache simulates unlimited direct access storage space. Data exchanges to and from the underlying hierarchical storage manager (HSM) are performed automatically and invisibly to the user. File system namespace operations may be performed through a standard NFS interface allowing all regular file system operations except accessing the data directly. In addition to standard data access methods like FTP, GridFTP, and http, the native access protocol dCap may be used allowing POSIX file system operations. dCache has full control of the location and multiplicity of datasets. Non-precious files are removed if space is running short. File replicas are generated if a certain pool becomes overloaded. Replicas are slowly removed if the situation improves. Pools are chosen for file transfers, either from clients or from the backend HSM, based on dynamic space and load parameters of the individual pools. In addition to the dynamic behavior, pools can be assigned to data according to the IP address of the client, the ordering mechanism of the backend HSM or special tags which can be given to subdirectory trees of the file space. For a detailed description of the scope of the dCache project see http://www.dcache.org. For more information on installation, configuration, administration and security considerations there is a dCache guide, “dCache, the book”, available (see: http://www.dcache.org/manuals/Book/).

Because dCache can be used in a local environment as well as over a Wide Area Network, firewall issues have to be considered as well. If components of the dCache system are distributed across multiples sites, some of these components have to be accessed from outside, which implies that firewalls have to be traversed. Protocols used by dCache are dCap, GSIdCap, GridFTP and SRM (storage resource manager).

dCap should be used only for local, trusted access; therefore it is not of any relevance for firewall considerations. GSIdCap extends the dCap protocol by using a GSI authentication wrapper (tunnel). Communicating with the GSIdCap servers (door nodes) requires opening certain ports into a firewall.

The GridFTP protocol will be described in a following chapter.

The SRM protocol uses https as transport protocol and negotiates data transfers between the client and server as well as between different servers. One of the other already mentioned protocols is used for the actual data transfer.

A common solution to overcome the problem of dynamic client and server connections over ports not known in advance is to open a range of ports within the firewall. From the user's perspective this allows undisturbed usage of dCache services. From the firewall manager's perspective it implies a security hole within the security policies of the site.

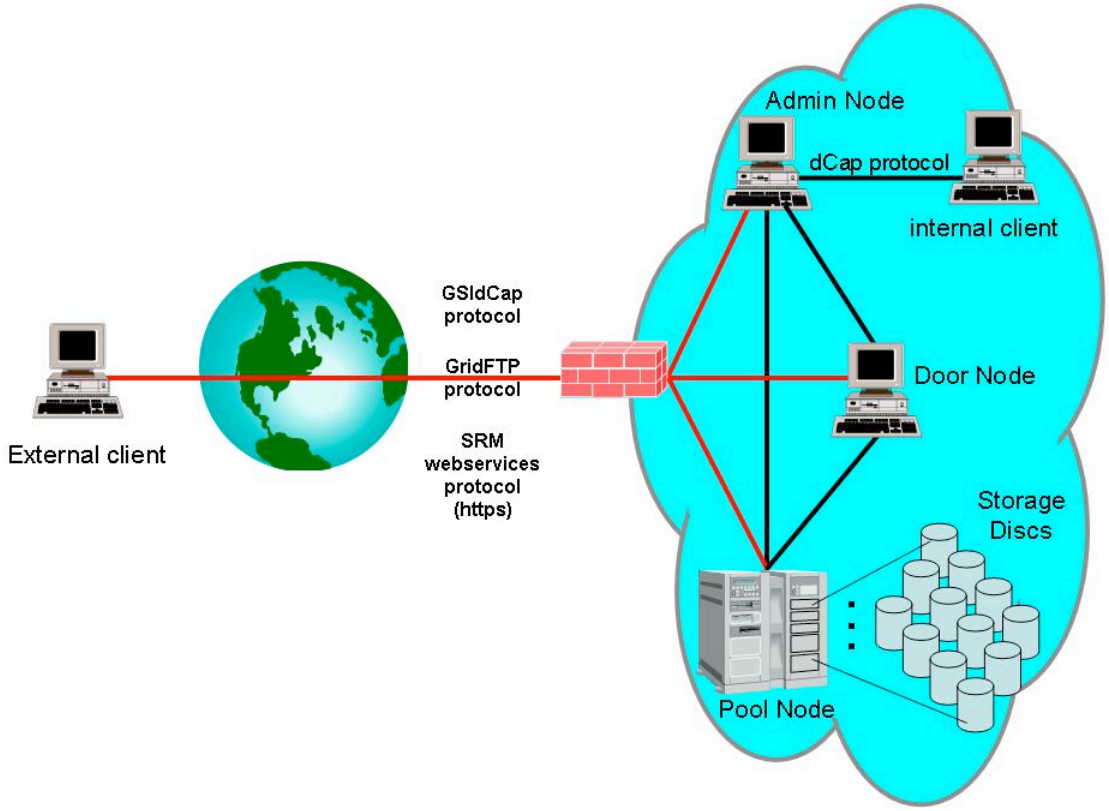


Figure 3: dCache, an overview, Nicolo Fioretti, Bari , Nov. 2005, http://www.dcache.org/manuals/dcache.nicolo.overview.small.jpg

3.2.3 Issues in enabling General Parallel File System, GPFS

Organization: Forschungszentrum Jülich GmbH, Jülich, Germany

The General Parallel File System, GPFS, developed by IBM, is a high-performance shared-disk file system. It provides fast, reliable data access from all nodes in a homogenous or heterogeneous cluster running AIX or LINUX operating systems.

GPFS allows parallel applications to simultaneously access one file or a set of files from any node that has mounted the GPFS file system while providing a high level of control over all file system operations. GPFS has been designed to deliver high performance, scalability and failure recovery by accessing multiple file system nodes in parallel. Nevertheless it complies with normal UNIX file system standards. GPFS provides high-performance I/O by "striping" blocks of data from individual files across multiple disks (on multiple storage devices) and reading/writing these blocks in parallel. In addition, GPFS can read or write large blocks of data in a single I/O operation, thereby minimizing overhead. For optimal performance and reliability the data can flow between the storage and the application node via multiple paths. GPFS availability is further improved by automatic logging and replication. Additionally GPFS can be configured to failover automatically in the event of disk or server malfunctions. GPFS scalability and performance are designed to meet the needs of data-intensive applications such as engineering design, digital media, data mining, financial analysis, seismic data processing and scientific research (see also the IBM GPFSflyer072606 “General Parallel File System” from July 2006 at http://www-03.ibm.com/servers/eserver/clusters/software/gpfs.html).

The general communication scheme used by GPFS is a client server model. The GPFS daemon (mmfsd process) communicates between nodes in different clusters. The communication paths are established via TCP socket call. GPFS uses IANA assigned port 1191 by default, which is changeable via the *mmchconfig* command if required. So from a firewall perspective GPFS uses only one port. This can be configured without any problems in standard firewalls. Because systems using GPFS are known in advance, a static access list can be configured. In future problems could arise if GPFS becomes publicly available and commonly used. In this case the protocol itself would have to be analyzed and secured, so that no backdoors or vulnerabilities could open holes within normally strongly protected network areas.

What makes GPFS interesting as a special firewall use case is its very high communication throughput. Because of parallel streams transferring a file between systems, high bandwidth is needed. Communication throughput is only dependent on the number of clients and I/O servers employed within the GPFS installation. Data rates of 3 GB/s have already been experienced. This implies the usage of high-speed firewalls, not yet available today, or a very good load balancing of firewall clusters.

3.3 System Deployment

3.3.1 The Issue with the “Net of Trust” and the “Bastion hosts” solution

Organization: Forschungszentrum Jülich GmbH, Jülich, Germany

The Research Center Jülich has been involved in many networking projects over the last 10 years. These projects always included research on new network technologies as well as their impact on applications. As a consequence firewall considerations have always been of main interest.

We realized that constantly growing network bandwidth demands require a reconsideration of the underlying techniques. New generation networking in Wide Area Networks implies the communication between hosts between different administrative security domains. Because of the high-speed network requirements, it is not possible to inspect every packet. Moreover firewalls cannot be faster than normal network interfaces (as they use these interfaces), so there will always be a time delay in implementing faster firewalls. Because firewalls have to forward many communication streams in parallel providing access for many different host-to-host communications, this scenario increases the needed throughput bandwidth enormously. Therefore traditional firewalls cannot be used in futuristic scenarios. How can security issues be handled in the future?

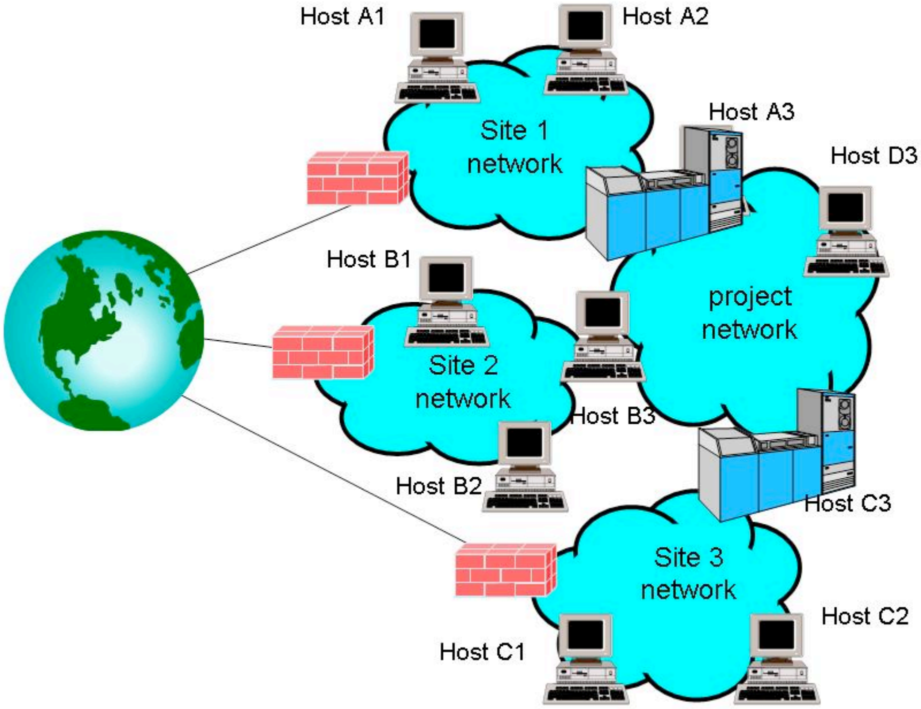


Figure 1: Securing project networks – “Net of Trust”

A generally used approach is to deploy a “Net of Trust”, which implies every node within the project network is assumed to be secure (or conforms to the security policies of all organizations). This can be achieved by having each node secured by an organization firewall (site local firewall) prohibiting unauthorized access from remote sites and assuming that only authorized persons can access the project network directly. Hosts A3, B3, C3 and D3 are connected to their institution networks and additionally to the project network. The project network cannot be accessed directly from outside (see Figure 1 above).

Alternatively each node within the project network can be installed with the highest security considerations (host firewall, iptables, virus scanners, only essential services installed and activated with minimum privileges, etc.).

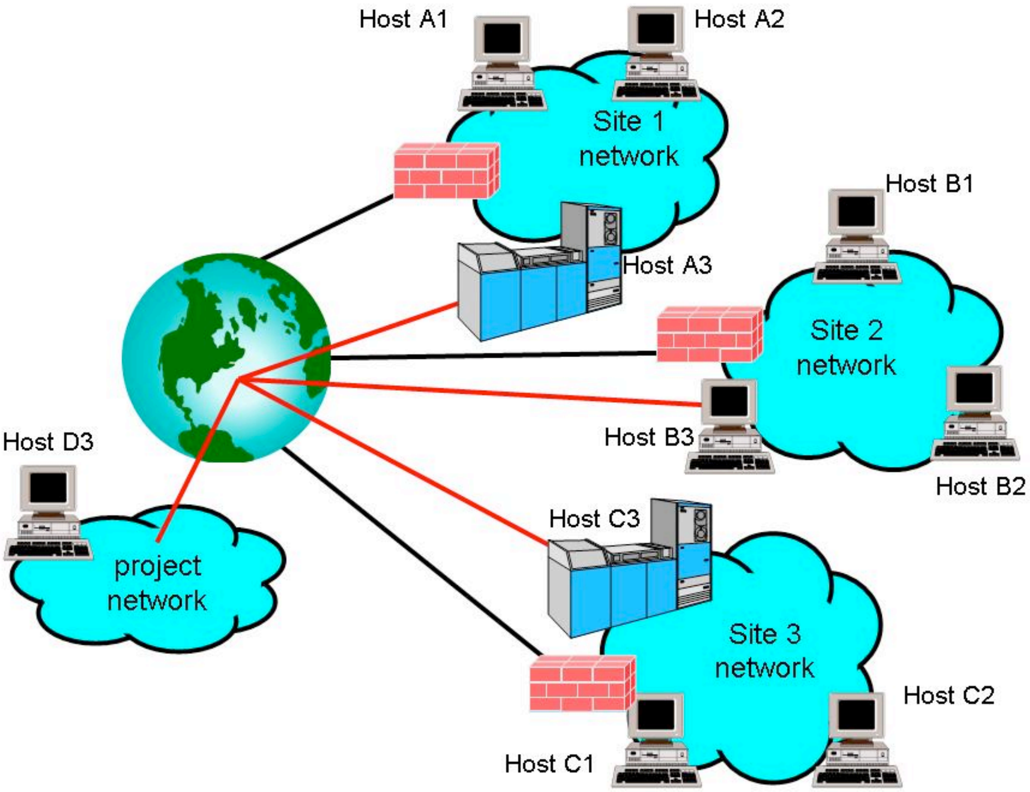


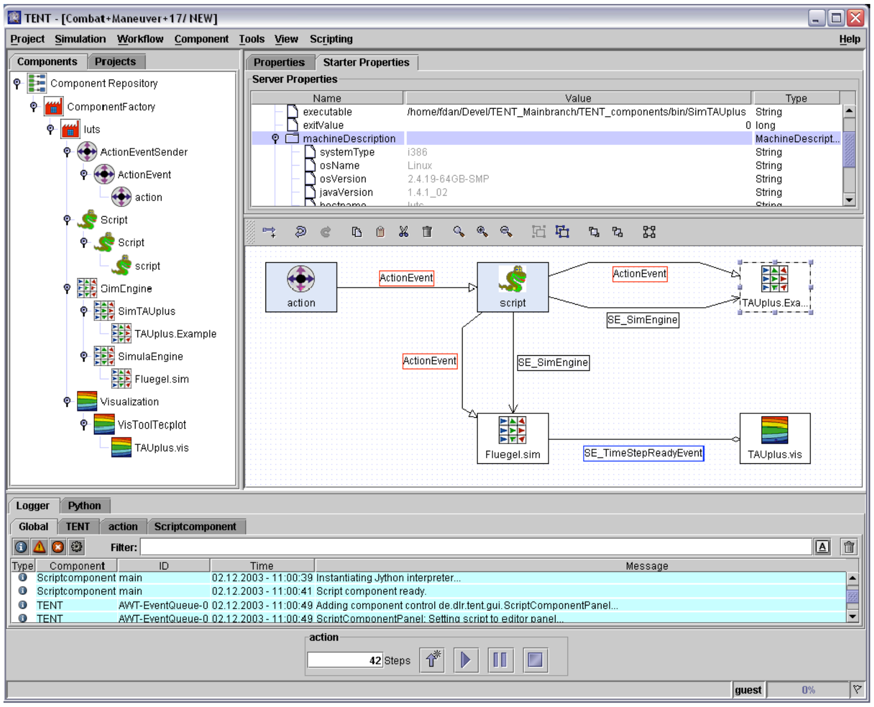
Figure 2: Securing project networks – "Bastion hosts"

These hosts are normally called bastion hosts, because they are located in an insecure environment and have been secured as a bastion against their enemies. Figure 2 shows the bastion host scenario, where hosts A3, B3, C3 and D3 are connected to their institution network as well as to the publicly accessible project network. All hosts within the project network have to be secured accordingly. Though these scenarios are a nightmare for firewall administrators and security officers they are often used because of missing alternatives. New ideas have to be developed in the future.

3.3.2 The workflow management system TENT

Organization: German Aerospace Centre, Cologne, Germany

The workflow management system TENT (see figure 4 for a screenshot) has been developed at the German Aerospace Center over the last few years. It allows engineers to easily setup and to maintain workflows. Workflows are applications coupled together to form a process chain. Applications can be computational fluid dynamic (CFD) codes or graphical editors for visualization. Components can be numerical or functional units within a workflow, e.g. computational fluid dynamic (CFD) codes, graphical editors for visualization, or pre-/post-processors.



Name	Value	Type
executable	/home/fdan/Devel/TENT_Mainbranch/TENT_components/bin/SimTAUplus	String
exitValue	0	long
machineDescription		MachineDescript...
systemType	i386	String
osName	Linux	String
osVersion	2.4.19-64GB-SMP	String
javaVersion	1.4.1_02	String
hostname	luts	String

```
Type Component       ID                Time                    Message
     Scriptcomponent main              02.12.2003 - 11:00:39 Instantiating Jython interpreter...
     Scriptcomponent main              02.12.2003 - 11:00:41 Script component ready.
     TENT            AWT-EventQueue-0 02.12.2003 - 11:00:49 Adding component control de.dlr.tent.gui.ScriptComponentPanel...
     TENT            AWT-EventQueue-0 02.12.2003 - 11:00:49 ScriptComponentPanel: Setting script to editor panel...
```

42 Steps

Figure 4: TENT GUI

This process chain can be used to solve fluid dynamics, structural mechanics, and thermodynamic computations. Components are the smallest elements in a workflow and can be localized on distributed resources. Computational resources can be placed on different remote hosts. Coupling of resources can be achieved by means of Grid computing. By creating Virtual Organizations (VOs) it is possible to use all these resources as one. The following functionalities are necessary to use TENT within Grids:

- Access to all resources of a Grid. Authentication mechanisms need to be provided.
- Data transfers between the resources of a Grid must be possible. Data transfers can either be Reliable File Transfers (RFT) or status messages (MPI based messaging).
- Execution of CFD codes on Grid resources. Job Managers and their queues should be accessible to the TENT system.

For all these communications TENT uses service-based communication.

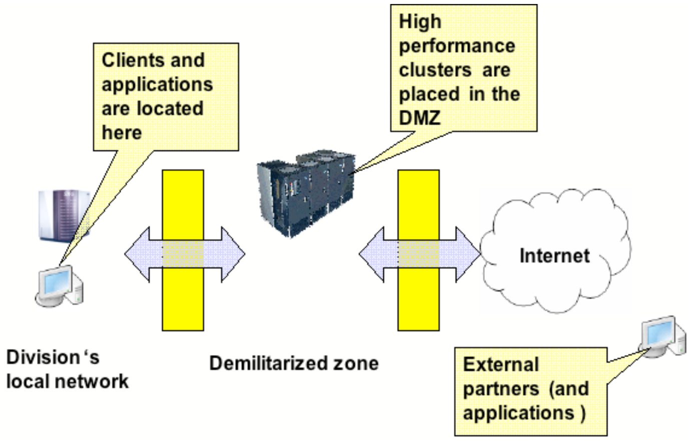

Clients and applications are located here
High performance clusters are placed in the DMZ
Internet
Division's local network
Demilitarized zone
External partners (and applications)

Figure 5: Closer look at possible firewall borders

The creation of VOs becomes obligatory when applications (and their matching licenses) and resources are located on either sides of a firewall. The creation of VOs can extend beyond the borders of companies. Therefore the location of the Grid resources is no longer bound to geographical positions. Figure 5 gives a closer picture of the borders of a company. Firewalls form the borders of the local site. But some applications and resources, like high performance cluster, are often not located within the local (and easily accessible) network.

Due to the fact that most companies and organizations use firewalls, the following problems may arise:

- Several firewalls have to be passed (internally and externally). The administrators of these firewalls are not always directly available.
- Firewalls have to be opened for several TCP and UDP ports. Some port ranges are unknown during set-up. They will be initialized by the Grid middleware itself. Consequently port ranges have to de defined.
- Data transfers have to be allowed beyond the borders of a local site. This includes the transmission of data packets and status information.
- VPNs have to be initialized at the borders of a site. To increase security the connections should be secured against wire-tapping.

Security policies refuse opening of firewalls in almost every case. Strict control of the incoming and outgoing traffic becomes a major issue. A lot of politics have to be dealt with when establishing connections beyond company's borders.

3.3.3 **AccessGrid**[1]

The Access Grid (see: "Welcome to AccessGrid.org" by Tom Uram, April, 20th, 2006, http://www.accessgrid.org/node/1) is a group collaboration system that provides resources as multimedia large-format displays, video conferencing systems, presentation and interactive environments, and interfaces to Grid middleware and to visualization environments, to allow group-to-group communication via the Internet. The Access Grid can be used for large-scale distributed meetings, collaborative work sessions, seminars, lectures, tutorials, and training.

[1] There is a concern that AccessGrid does not follow the traditional Grid Computing model, but the reason to include this case within this document is to address issues around use of multicast with firewalls.

Therefore it differs from desktop-to-desktop tools that allow only communications between individuals. The software is well accepted by the Grid community and widely used all over the world. So called Access Grid nodes are equipped with high-end audio and visual technology. Nevertheless, individual desktop users may participate in Access Grid communications too by installing software on the local host which provides one-to-many and one-to-one communications.

Access Grid users meet at virtual meeting spaces, which may be open to all or restricted to individual users. Each user connects to the Access Grid as an individual node, which may contain a desktop with a Quick Camera only or a highly equipped video conference room. Communication is established using multicast and consists of a number of independent parallel connections. This makes the overall communication stream difficult to manage within firewalls. Many firewalls do not allow or understand multicast traffic, which requires handling this application differently than others. Often, multicast traffic is bypassed or tunnelled through local firewalls, allowing only multicast addresses to use this “security shortcut”. Often this leads to conflicts with the existing institution’s network security policy and therefore requires management procedures to allow manual interaction/configuration on demand or general changes of security policies.

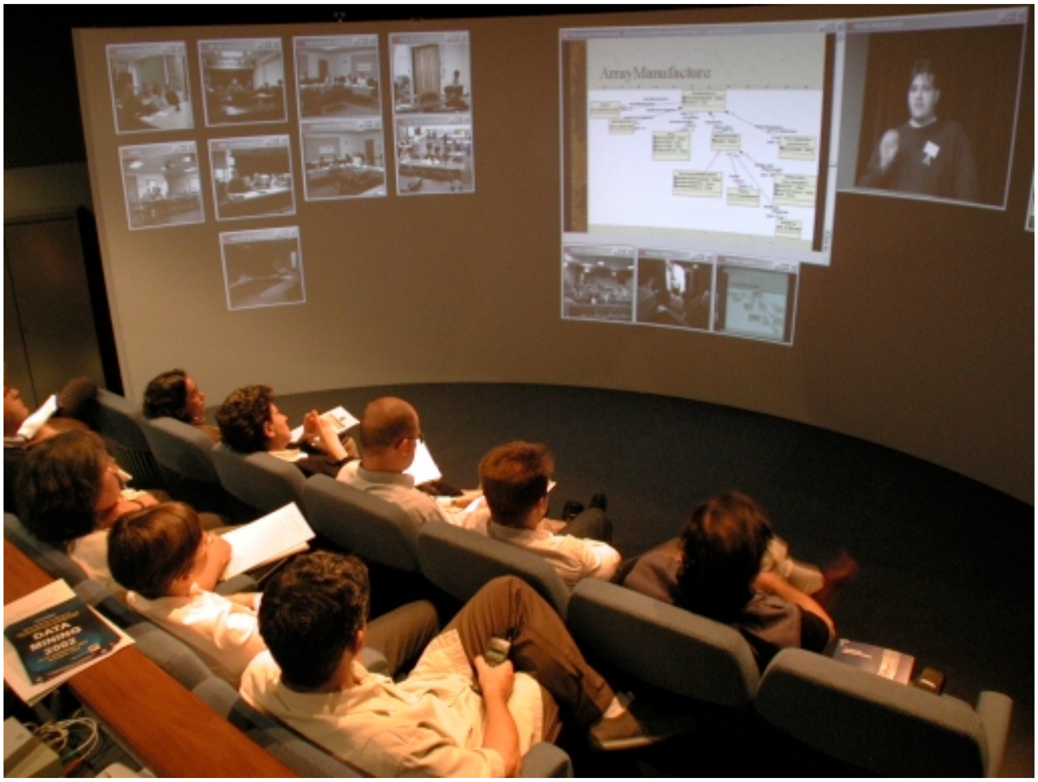


Figure 6: Example Access Grid conference

Access Grid communications include a number of different software tools for which firewall configuration have to be adjusted. Software tools include the Access Grid client software, inSORS IG client software, VIC/IG Video and RAT/IG Audio tools, Jabber client, tkMoo MUD (Multiple User Dimensions) client software and server software as IG PIX, VNC, DPPT (Distributed PPT). If you are using a client machine only, you can use a Multicast-Unicast bridge, where the client connects to the bridged video and audio ports. This allows avoiding multicast streams within Access Grid communications. A detailed discussion on communication streams initiated and ports used by Access Grid can be found at the Access Grid home page (see: “Access Grid Port Usage” by Javier Gomez Alonso, Univ. of Manchester, http://www.accessgrid.org/agdp/guide/ports/1.03/index.html).

3.3.4 Firewalls and high bandwidth, long distance networks

Organization: University of Amsterdam, Amsterdam, The Netherlands

Grid applications often use high-bandwidth connections between grid locations over long distances. These applications will benefit from congestion-free connections. A modified TCP protocol behavior, which increases the rate of transmissions more rapidly after a congestion event, is needed to efficiently use such a connection. Such behavior makes these TCP streams unsuitable to share bandwidth with regular TCP streams, as they are considered to be unfair. These TCP streams therefore typically by-pass the regular Internet using dedicated, mostly optical-, connections between grid locations. Research within the GHPN-RG is performed to create on-demand version of these connections, using switched optical network technologies. The GHPN group does not consider the involved network security architectures.

This section considers the requirements towards a possible security architecture that could be used to connect a grid node both to the Internet and to a long distance by-pass network.

The figure below shows a possible network layout involving firewalls. All Grid resources are located behind a typical two-firewall setup with a DMZ. Firewalls A and D have an additional network interface that connects to the high bandwidth connection. Involving Grid middleware, a grid application may schedule a connection via the Multi-domain control and management plane. A Grid VO may be involved in the decision to provision the connection.

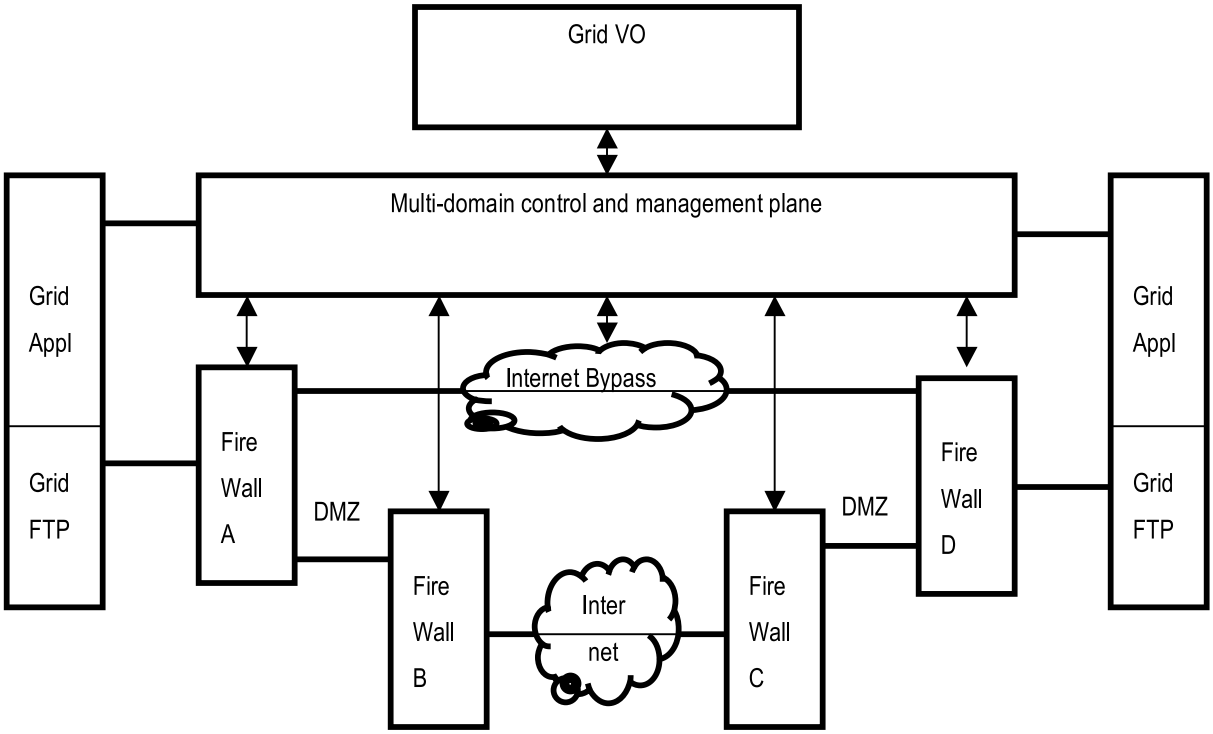


Figure 12: Possible network layout with a typical two firewall-setup with a DMZ

Consider a network architecture as above, the following requirements and issues can be defined:

1. The grid location must be protected against malicious attacks from the Internet. This is a general requirement for any node connected to the Internet.
2. The Internet must be protected from abusive use from a Grid cluster. As Grid locations are capable of generating vast amounts of IP traffic, it has the potential to attract malicious users. If access control fails, then a firewall should stop any attempt to misdirect IP traffic.
3. Memory shortage in any forwarding device will cause packet-loss. At least the performance of firewalls A and D must allow wire-speed operation without any congestion. Both firewalls must have enough memory to support the bandwidth delay product for each by-pass connection. Note that long distance connections inherently have large bandwidth-delay products.
4. Firewall architectures must be capable of splitting the high-bandwidth inter-grid location traffic from the regular traffic at a very early stage, e.g. at firewalls A and D. This will avoid high network loads at shared network resources downstream.
5. Adding nodes to a grid cluster should scale with the firewall infrastructure, such that congestion for Grid TCP streams, and unfairness to regular Internet resources, is avoided.
6. Usage of on-demand network resources between two Grid locations may need authorization. Firewalls A and D may act as an ingress/egress point of such a connection. Firewalls A and D could therefore act as an access enforcement point. Firewall A or D could act as a first point of contact to which applications could send requests to open up a chain of firewalls.
7. Firewalls A and D should enforce that only private (non-routable) addresses can be used via the by-pass connection. Routable addresses should be forwarded via the DMZ to firewalls B and C. Some address management for the private address spaces must be performed.
8. Both, the Grid VO (or users authorized by the Grid VO) and network security administrator should have a stake in the control of the firewalls A and D.

4 Classification of Firewall Issues

Given that:

- Chapter 4 presented several kinds of grid applications, showing the issues arising when firewalls are located within the communication path.

- Appendix A describes the issues of each specific application in a more structured way, defining the following classes of problems:
  - Software
  - Hardware
  - Network
  - Security policy

This chapter will recognize two categories of firewall issues and further subdivide them accordingly. Issues are either caused by:

1. The fact that the application is unaware of the network.
2. The fact that the network is unaware of the application.

The classification is made based on the assumption that both the network and the application may not be aware of each other in terms of requirements. Typically, the application assumes network transparency. However, the application also expects secure and reliability operation and therefore expects to be protected against malicious intends. Both expectations require some understanding between the application and the network. We therefore approach the classification from the above listed observations. At this point we do not seek solutions for these issues, we sometimes only hint towards them. Presenting existing and new solutions will be covered in (a) separate document(s).

4.1 Issues caused by the application having difficulties to be aware of network needs

This is an issue were applications try to adapt towards the needs of the network. This paragraph tries to map each firewall issue as identified in chapter 4 in one of the following four categories: software, hardware, network, security policy.

4.1.1 Software and port numbers

Port numbers and number of ports are unknown until the application starts. The consequence is that firewall administrators need to create big holes (up to 10.000 ports) if the application is not capable of determining the amount of ports to be used and/or the port numbers are unknown. Trying to push all traffic though a single hole (e.g. HTTP port 80) causes referral problems. In general, only specific, predetermined applications that use a low number of very well-defined ports (or “well-known ports”) can be supported adequately.

4.1.2 Hardware

Applications that want to be aware of the underlying network have difficulties with:

- Understanding the number and kind of firewalls located within the routing path.
- Pushing high performance data streams across long connections that need enough buffer space and switching capacity. If applications were aware of buffer sizes and delays, they

could adjust their transmission rates more effectively to avoid packet drops in any kind of forwarding device, including firewalls.
- Opening multiple high performance channels (wavelengths) over a single fiber. There are no firewalls that are able to deal with multiple wavelengths on a single fiber. If these wavelengths have been divided into individual fibers by DWDM equipment, firewalls are not able to deal with 16, 32 or 64 links of 10 Gb/s each currently. Current firewalls can deal with up to 5 Gb/s links, and, if they act as packet filters only, may handle multiple 10 Gb/s links, but they are not able to deal with several hundred Gb/s coming in through multiple 10 Gb/s interfaces. Though load balancing firewalls are available since some time, these cannot handle such high communication streams.

4.1.3 **Network**

Applications are typically unaware of their position within the network. This may cause issues like:

- Certain grid applications cannot be placed inside the DMZ. This, as the data contained within such application may be too sensitive to allow it to be compromised and therefore can only be placed within the enterprise network. The application will need to be changed such that it temporally publishes relevant pieces of the information from its location on the enterprise side of the firewall to a server reachable via the DMZ.
- Grid applications are more and more developed using a SOA. Such architecture is inherently distributed. If a workflow orchestrates components located at various places, the interfaces may need to cross multiple firewalls and DMZs, each with their own security and firewall policies. The more a workflow is allowed to be flexible, the more security policy issues are likely.
- Applications are built independently of their network addresses, but rather have things like URLs to identify them. Applications with a need for special network resources, that bypass the regular Internet, must somehow indicate this. Therefore, firewalls involved in bypass connections also may need to perform elaborate routing functions,

4.1.4 **Security Policy**

In terms of security the application may need to communicate certain needs to the network. This causes typically issues like:

- Firewalls may not have enough information to authorize complex grid applications.
- Firewalls must not only protect against attacks from the public network, but also prevent the public network from being abused where the application does not provide enough information to distinguish between “good” and “bad”.
- Applications need to trust each other and firewalls may not be able to extend the security context between two applications.
- Applications cannot provide firewalls with enough information so that firewalls may therefore not be aware if a host connecting is actually trusted.

4.2 Issues caused by the network being unable to be aware of the application

These issues are cause by problems where the network tries to adapt towards the need of an application. This is the traditional approach where the application expects the network to be transparent and the issue is therefore the network.

One may subdivide these issues into 5 kinds, with an increasing amount of difficulty to be resolved:

4.2.1 Applications that use only well–known TCP/UDP ports

These specific ports could be easily opened within a firewall and they should not represent any problem for a firewall administrator. Nevertheless, each of these applications has to be examined in detail, to verify whether the communication protocol complies with the security policy of the local organization. For example, ssh should be no problem, because users will always be authenticated and authorized by the local ssh server. The same would be true for telnet, but telnet uses an unencrypted authentication scheme, sending userids and passwords in cleartext over the communication path. If a telnet session is recorded by an unauthorized person (hacker), he has now gained access to sensitive information that may afterwards be used to get unauthorized access to resources that shouldn't be publicly available. So each grid application, although using only single and fixed ports, has to be checked for compliance with the organizational security policies.

Some applications extend the use of single well-known ports by tunneling messages through them e.g. port 80 (http). These implementations have been developed to circumvent institutional security policies exploiting the fact most organizations allow the use of the http protocol. Though this method would allow any kind of firewall traversal, it has been shown that, as a general concept, would also allow hackers and especially viruses to overcome firewall barriers. Firewall developers have taken application tunneling into account and have developed countermeasures against these bad-practices. They have implemented software that is familiar with a set of standard protocols, e.g. http, making a trivial port tunneling to be recognized and stopped. This remedy does not completely solve the problem, because the tunneling application can be programmed to behave like a standard protocol stream, but it helps in recognizing most of the trivial tunnel-attacks from the outside. Taking into account these considerations, application programmers should avoid making use of these kinds of tunneling techniques.

4.2.2 Detectable dynamic data transfer ports

Applications that use a single well-known port for a control channel and a set of dynamic ports for the data transfer. The control channel (typically in clear-text) is used to synchronize the communication behavior between client and server applications, e.g. to exchange information about the dynamic ports that will be allocated for file transfers (data stream of an ftp session).

The control streams can be constantly monitored by special firewall plug-ins that extracts the set of ports dynamically allocated for the data streams. These ports are then automatically opened in the firewall. Such a mechanism has been already developed for the FTP, H.323 and SIP protocols.

4.2.3 Obscured dynamic data transfer ports

Applications similar to those described in bullet 2, but where the information exchanged in the control channel is not sufficient to determine the set of dynamic ports that will be used for data transfer. This may happen because the control channel is encrypted (thus the firewall plug-in cannot access its content) or because the set of dynamic ports is generated internally by the application after the control channel has been closed (e.g. GridFTP 3rd party transfer). In both cases the applications require a complete range of ephemeral ports to be permanently allowed for traversing through the firewall. Most likely, firewall administrators do not like such a configuration.

4.2.4 Arbitrarily dynamic data transfer ports

Applications that change on-the-fly the total amount of dynamic ports they use. These applications may start a data transfer using one ephemeral port or a set of ephemeral ports, then they may add or remove sockets arbitrarily at any time (i.e. GridFTP 3rd party transfer).

The difficulty in determining the complete communication setup at starting time implies that a complete range of ephemeral ports need to be permanently allowed for traversing through the firewall. Most likely, firewall administrators do not like such a configuration.

4.2.5 High throughput data pipes with non standard traffic patterns

Applications that require high throughput data pipes. These data streams often have special SLAs (Service Level Agreements) that make them unsuitable to share communication links with normal traffic. These SLAs could result in unfair behavior of the streams, leading to reduced throughput for normal traffic. In some cases normal traffic could drop to zero because of the excessive use of these grid applications.

As a solution it should be feasible to route the special traffic on dedicated connections, bypassing the normal institutional firewall, while yet ensuring full compliance with the security policies (high throughput traffic has to be secured in a different manner).

5 Future directions

Firewalls aim at securing and controlling the traffic flowing in and out of an organizational domain. The importance of their role in the enforcement of security policies is not under discussion. On the other hand, free research and information exchange between organizational entities is required as well.

Application programmers did not deal with firewalls in the past. Applications were often developed, debugged and validated in a local environment without interaction with firewalls. After successful implementation they have been thought to be deployed in a more global environment, often extending over different organizational entities. In this way applications and firewalls came to interact with each other, establishing a relationship not always easy and flawless. The examples gathered in this document have the purpose to give programmers useful insights for developing firewall-aware applications, increasing their awareness of the distributed nature of resources of a grid environment.

Another line of investigation should pave the way for firewall developers to produce new kinds of firewalls, which can cope with new types of applications and network infrastructures. Constantly growing bandwidth demands require reconsideration of the underlying technologies. Firewalls cannot be faster than network interfaces since they are based on these interfaces; there will always be a delay in implementing faster firewalls. New concepts have to be developed: for example instead of inspecting single packets streams could be checked. This is already offered by current firewalls through the port concept. Many connections will be allowed without checking their actual content. The connections will be allowed because another entity, the destination system, already verified the authorization credentials.

Firewall vendors should also get involved in accomplishing strategic and long-term objectives, as the definition of a standardized authentication/authorization mechanism to be implemented in their systems. Such a protocol would allow grid-enabled firewalls to become a reality.

6 Summary

This document starts with an introduction of the fundamentals about firewalls and other devices, like network address translators, application level gateways and VPN gateways, used to protect applications and infrastructure from malicious attacks from the Internet.

The next section gives an overview of some commonly deployed grid applications and the issues they face when dealing with firewalls. It is not intended to include all possible applications used in grid environments, but just to identify and describe a set of representative examples. These applications were classified according to their communication behavior, to get a good description of the problems arising because of the existence of firewalls within the communication paths. To perform an objective analysis of the firewall issues, based exclusively on homogeneous parameters, the chosen applications were described according to a template that accommodates four categories of problems/risks: software, hardware, network, and security policy. The complete set of descriptions is available in Appendix 1.

The last part of the document focuses on the identification of regular patterns among the firewall issues. Most often problems arise because of the large number of ports used by the applications or because of the impossibility to determine all these ports in advance. Other common shortcomings are caused by hardware limitations or by specific network configurations, like the deployment of grid services in the DMZ.

The OGF Firewall Issues Research Group (FI-RG) intends to create another document as a follow-up to this one, which will introduce possible approaches to solve (or mitigate) the problems identified so far.

7 Security Considerations

This entire document is about security considerations.

It describes grid applications used across firewalls, tries to identify security risks and organizes these risks into use cases. The document is intended to provide an overview of scenarios not yet supported by current firewall systems and aim at identifying solutions for future developments.

8 Acknowledgements

The authors whish to thank Raju Shah (Force10), Dave Wesner (Research Center Jülich) Melinda Shore (Cisco) for feedback and comments on the document as well as for proof reading, corrections of spelling, grammar and style. We also would like to acknowledge the presentations from researchers in the FI-RG sessions that helped shape this document.

Also we would like to thank the OGF security area directors Olle Mulmo and Dane Skow for supporting our work.

9 Contributors

Ralph Niederberger (Editor)
Forschungszentrum Jülich GmbH
r.niederberger@fz-juelich.de

William, E. Allcock
Argonne National Laboratory
allcock@mcs.anl.gov

Leon Gommans
University of Amsterdam
lgommans@science.uva.nl

Egon Gruenter
Forschungszentrum Jülich GmbH
e.gruenter@fz-juelich.de

Thijs Metsch
Deutsches Zentrum für Luft- und Raumfahrt - DLR e.V.
thijs.metsch@dlr.de

Inder Monga
Nortel Networks
imonga@nortel.com

Gian Luca Volpato
RRZN – Leibniz Universität Hannover
volpato@rrzn.uni-hannover.de

Christian Grimm
RRZN – Leibniz Universität Hannover
grimm@rrzn.uni-hannover.de

10 Glossary

ALG	Application Level Gateway, see chapter 3.3
dCAP	dCache native protocol providing access to dataset contents and supporting regular file access functionality. The dCache software package includes a C-language client implementation of this protocol offering the POSIX file I/O operations as well as the standard file system namespace operations.

DMZ	Demilitarized Zone. DMZ is a firewall configuration for securing local area networks (LANs). In a DMZ configuration, most computers on the LAN run behind a firewall connected to a public network like the Internet. One or more computers also run outside the firewall, in the DMZ. Those computers on the outside intercept traffic and broker requests for the rest of the LAN, adding an extra layer of protection for computers behind the firewall. Traditional DMZs allow computers behind the firewall to initiate requests outbound to the DMZ. Computers in the DMZ in turn respond, forward or re-issue requests out to the Internet or other public networks. The LAN firewall, though, prevents computers in the DMZ from initiating inbound requests.
DWDM	Dense Wavelength Division Multiplexing is a fiber-optic transmission technique that employs light wavelengths to transmit data parallel-by-bit or serial-by-character.
GPFS	General Parallel File System, developed by IBM, is a high-performance shared-disk file system. It provides fast, reliable data access from all nodes in a homogenous or heterogeneous cluster running AIX or LINUX operating systems.
GridFTP	Special FTP protocol for Grids, see chapter 4.2.2
GSI	Grid Security Infrastructure, the basis for Globus Toolkit Security layer.
GSIdCap	Extension of the dCap protocol using GSI authentication wrapper (tunnel). Communicating with the GSIdCap servers (door nodes) requires opening ports into a firewall.
HSM	Hierarchical Storage Manager is policy-based management of file backup and archiving in a way that uses storage devices economically and without the user needing to be aware of when files are being retrieved from backup storage media. Although HSM can be implemented on a standalone system, it is more frequently used in the distributed network of an enterprise. The hierarchy represents different types of storage media, such as redundant array of independent disks systems, optical storage, or tape, each type representing a different level of cost and speed of retrieval when access is needed.
H.323	H.323 is an umbrella recommendation from the ITU-T that defines the protocols to provide audio-visual communication sessions on any packet-switched network.
IDB	Incarnation Database
IPSec	**IP Sec**urity, a set of protocols developed by the IETF to support secure exchange of packets at the IP layer. IPsec has been deployed widely to implement Virtual Private Networks (VPNs).
MAC	Medium Access Control. This protocol is used to provide the data link layer of the Ethernet LAN system.
MPI	Message Passing Interface. MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users.
NAT	Network Address Translation, see chapter 3.3

NAPT	Network Address Port Translation, see chapter 3.3
NJS	Network Job Supervisor. The NJS is the Unicore front-end to a target batch system. It is responsible to map the abstract job description to concrete target system instances.
PAT	Port Address Translation, see chapter 3.3
RFT	Reliable File Transfer. It is an OGSA-based service that provides interfaces for controlling and monitoring third-party file transfers using GridFTP servers. The client controlling the transfer is hosted inside of a Grid service so it can be managed using the soft state model and queried using the ServiceData interfaces available to all Grid services.
SIP	Session Initiation Protocol. It is an application-layer control protocol that can establish, modify, and terminate multimedia sessions such as Internet telephony calls (VoIP). SIP can also invite participants to already existing sessions, as in multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility - users can maintain a single externally visible identifier regardless of their network location. See also RFC 3261, 3262, 3263, 3264, and 3265.
SLA	Service Level Agreement. It is a formal contract between a carrier and a customer that defines the terms of the carrier's responsibility to the customer and the type and extent of remuneration if those responsibilities are not met. These agreements can handle e.g. latencies, packet loss, cable damage etc.
SOAP	Simple Object Access Protocol. It is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.
SOCKS	Socks is a protocol for secure sessions traversal across firewall. It provides a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall. See RFC 1928.
SRM	Storage Resource Manager. This protocol uses HTTPS as transport protocol and negotiates data transfers between clients and servers as well as between different servers.
SSL	Secure Socket Layer is an encryption standard that prevents anyone from intercepting and reading the data streams between the clients and servers
TSI	Target System Interface. It is a library of Perl modules being installed on the target system (e.g. the supercomputer)
TTL	Time-to-live. It is a value in an Internet Protocol packet that tells a network router whether or not the packet has been in the network too long and should be discarded. For a number of reasons, packets may not get delivered to their destination in a reasonable length of time. A solution is to discard the packet after a certain time and send a message to the originator, who can decide whether to resend the packet.
VLAN	Virtual LAN. It is a group of PCs, servers and other network resources that behave as if they were connected to a single, network segment — even

	though they may not be. The resources and servers of other users in the co-location facility will be invisible to each of the other VLAN members.
VPN	Virtual Private Network. It is a private network that is configured within a public network (a carrier's network or the Internet). VPNs are widely used by enterprises to create wide area networks that span large geographic areas, to provide site-to-site connections to branch offices and to allow mobile users to dial up their company LANs.
XML	Extensible Markup Language. It is a simple dialect of Standard Generalized Markup Language (SGML). Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.
X.509	X.509 is a widely used standard for digital certificates.

11 Appendix 1: Classification of firewall issues seen from the use cases side

Name	The “Net of Trust Model”					
Description	Hosts within a project network spanning different organizational entities are secured via institutional firewalls. Between the project hosts no firewall is used. Each host and the users of these hosts are assumed trustworthy. Advantage and problem solved: Because of private networks, firewalls do not introduce a throughput bottleneck (10 Gb/s and more connections may be used)					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		low	low	High
Occurrence		NA		NA	management	management
	No elements within communication path.	**Own Software**	Any kind of software can be used. Commercial, free software as well as experimental software.	No hardware restrictions. Because of free communication paths every kind of hardware using any kind of protocols (also non IP) may be used.	The network connecting the hosts is a private one. Could be IP or lower protocols.	The security policy on both sides has to agree with this net of trust concept. Hacking of one project host leads to security impacts on all connected institutional local networks.
		Ports used	Because of no restriction, every port/port range may be used.			
		Protocol used	All kinds of protocols beneath TCP and UDP possible			

Name	The “Bastion Host Model”					
Description	Hosts within a project network spanning different organizational entities are secured only by their own security mechanisms (host firewalls). The project hosts are freely accessible from the outside world. The project network security concept is based on the security of each individual host (bastion host).					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		low	low	High
Occurrence		NA		NA	management	management
	No elements within communication path.	**Own Software**	Any software can be used assumed this software packet is secure and does not introduce any vulnerability.	No hardware restrictions. Because of free communication paths every kind of hardware using any kind of protocols (also non IP) may be used. Prerequisite: Host can be configured secure (whatever this means).	The network connecting the hosts is an official one. Could be IP or lower protocols.	The bastion hosts are placed outside the institution networks. This implies that these networks are not affected. Nevertheless, connections from the bastion hosts into the institution network are normally required. These communications have to be inspected and secured. Hacking of a project host does not directly lead to security impacts on the other project hosts. Every host is a standalone bastion.
		Ports used	Because of no restriction, every port/port range may be used.			
		Protocol used	All kinds of protocols beneath TCP and UDP possible			

Name	Access Grid					
Description	The Access Grid is a group collaboration system that provides resources as multimedia large-format displays, presentation and interactive environments, and interfaces to Grid middleware and to visualization environments, to allow group-to-group collaboration via the Internet. The Access Grid can be used for large-scale distributed meetings, collaborative work sessions, seminars, lectures, tutorials, and training. Well accepted software within the Grid community and widely used all over the world.					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		Low	high	high
Occurrence		NA		NA	management	management
	Any kind of firewalls between the communicating entities.	**Own Software**	No. Software toolkit developed by Argonne National Lab, ANL	Any kind of hardware as desktop systems, video equipment, …	Allowing multicast, bypassing firewalls or configuring tunnels through firewalls.	Often conflicting with the existing institution's network security policy. Requires management procedures to allow manual interaction/configuration on demand or general changes of security policies (opening multicast traffic, tunnelling through firewalls or bypassing firewalls.
		Ports used	**Incoming and Outgoing**: many different fixed ports depending on used software components			
		Protocol used	Multicast			

Name	dCache					
Description	dCache allows storing and retrieving huge amounts of data, distributed among a number of heterogeneous server systems. These systems simulate a single virtual file system.					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		low	middle	High
Occurrence		NA		NA	management	management
	Any kind of firewalls between the communicating entities.	**Own Software**	No. Software developed at DESA and FERMI.	No hardware restrictions.	Different kinds of configuration allowed. Some components must/may be placed within a DMZ, some of them must/may be placed internally into the site network.	Since most of the protocols use dynamic ports within a specified range, there have severe security impacts. If the protocols used haven't been configured securely, backdoors may be introduced.
		Ports used	**Incoming**: dCap TCP 22125 GSIdCap TCP 22128 GridFTP TCP 2811 and 20000-25000 SRM TCP 8443 Location Manager TCP 11111 **Outgoing**: any *All ports are configurable*			
		Protocol used	TCP			

Name	GPFS					
Description	The General Parallel File System is a high-performance shared-disk file system. It provides fast, reliable data from all nodes in a homogenous or heterogeneous cluster running an AIX or LINUX operating system. GPFS allows parallel applications simultaneous access to one file or a set of files from any node that has the GPFS file system mounted using parallel streams for a single file transfer.					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		low	low	Middle
Occurrence		NA		NA	NA	management
	Any kind of firewalls between the communicating entities.	**Own Software**	No. Software developed by IBM.	No hardware restrictions.	Communication is done via normal communication paths. (Site network – provider network – site network).	Protocol uses fixed configurable TCP port. Disadvantage: Communication including data is unencrypted.
		Ports used	GPFS TCP 1191 *Port is configurable*			
		Protocol used	TCP			

<table>
<tr><td><b>Name</b></td><td colspan="6">The workflow management system TENT</td></tr>
<tr><td><b>Description</b></td><td colspan="6">This use case describes the firewall issues arise while integrating grid middleware software into the workflow management system TENT. The creation of a VO forms the major problem.</td></tr>
<tr><td></td><td><b>Elements in communication path</b></td><td colspan="2"><b>Software</b></td><td><b>Hardware</b></td><td><b>Network</b></td><td><b>Security Policy</b></td></tr>
<tr><td><b>Severity</b></td><td></td><td colspan="2">Low</td><td>low</td><td>middle</td><td>High</td></tr>
<tr><td><b>Occurrence</b></td><td></td><td colspan="2">NA</td><td>NA</td><td>management</td><td>management</td></tr>
<tr><td rowspan="4"></td><td rowspan="4">Several packet filters located at the network borders of the participating organizations.</td><td><b>Own Software</b></td><td>Yes (TENT)
No (Globus Toolkit)</td><td rowspan="4">The hardware on which the software runs are not located in DMZs. Solutions with VPNs would end in the DMZ. Resources of the Grid cannot be relocated.</td><td rowspan="4">3 DMZs are located in the communication path.</td><td rowspan="4">The security policy on both sides does not allow the opening of ports without inspection.</td></tr>
<tr><td><b>Ports used</b></td><td>Unknown port range used</td></tr>
<tr><td><b>Protocol used</b></td><td>TCP</td></tr>
<tr><td colspan="2">Globus requires several ports to be opened for e.g. MyProxy Server, Web Service Container. GridFTP uses an unknown port range.</td></tr>
</table>

Name	The Globus Toolkit					
Description	This use case describes the firewall issues that arise while using Globus Toolkit applications					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		low	middle	High
Occurrence		NA		NA	management	management
	Several packet filters located at the network borders of the participating organizations.	**Own Software**	No (Globus Toolkit)	Runs on different kinds of Grid resources.	Runs on different kinds of Grid resources. Depending on the number of streams and the throughput desired using all kinds of Globus applications, the firewall hardware might be a performance bottleneck.	The security policy on both sides does not allow the opening of ports without inspection. Huge port ranges may have to be opened dependent on services used.
		Ports used	Unknown ports and known port ranges as well as known ports used			
		Protocol used	TCP			
		Globus requires several ports to be opened for different services. To be highly parallel for some applications huge port ranges have to be opened				

<table>
<tr><td><b>Name</b></td><td colspan="6">GridFTP vs. the Firewall</td></tr>
<tr><td><b>Description</b></td><td colspan="6">GridFTP protocol specifics and the reason why firewalls are not able to deal with it well.</td></tr>
<tr><td></td><td><b>Elements in communication path</b></td><td colspan="2"><b>Software</b></td><td><b>Hardware</b></td><td><b>Network</b></td><td><b>Security Policy</b></td></tr>
<tr><td><b>Severity</b></td><td></td><td colspan="2">low</td><td>low</td><td>low</td><td>High</td></tr>
<tr><td><b>Occurrence</b></td><td></td><td colspan="2">NA</td><td>NA</td><td>NA</td><td>management</td></tr>
<tr><td rowspan="4"></td><td rowspan="4">Unknown number of Packet filters/ stateful firewalls monitoring based on 5-tuple of an IP packet</td><td><b>Own Software</b></td><td>Yes - GridFTP</td><td rowspan="4">Runs on Grid resources. Grid resources cannot be placed in the DMZ</td><td rowspan="4">Runs on Grid resources. Depending on the number of streams and the throughput desired using GridFTP, the firewall hardware might be a performance bottleneck.</td><td rowspan="4">Requires static opening of a large number of ports (1000+ at least) in the dynamic port range all the time in Firewall. This leads to a big security hole that security and network administrators are challenged to endorse.</td></tr>
<tr><td><b>Ports used</b></td><td>Unknown numbers / dynamically decided</td></tr>
<tr><td><b>Protocol used</b></td><td>TCP</td></tr>
<tr><td colspan="2">Software requires multiple ports to run. Sockets/connections are added and deleted dynamically. Sockets determined dynamically per connection</td></tr>
</table>

<table>
<tr><td><b>Name</b></td><td colspan="6">UNICORE</td></tr>
<tr><td><b>Description</b></td><td colspan="6">The UNICORE software (UNiform Interface to COmputing REsources) is a user-friendly software interface which allows easy and uniform access to distributed computing resources, and which provides support for running important scientific and engineering applications in a Grid environment. Scientists can use different supercomputers as well as other computing and storage resources without having to become experts in the special kind of access software and security policies of the various (super-)computer centers.</td></tr>
<tr><td></td><td><b>Elements in communication path</b></td><td colspan="2"><b>Software</b></td><td><b>Hardware</b></td><td><b>Network</b></td><td><b>Security Policy</b></td></tr>
<tr><td><b>Severity</b></td><td></td><td colspan="2">Low</td><td>low</td><td>low</td><td>low</td></tr>
<tr><td><b>Occurrenc e</b></td><td></td><td colspan="2">NA</td><td>NA</td><td>NA</td><td>NA</td></tr>
<tr><td rowspan="3"></td><td rowspan="3">Any kind of firewalls between the communicating entities.</td><td><b>Own Software</b></td><td>Available via sourceforge.org</td><td rowspan="3">No hardware restrictions.</td><td rowspan="3">Communication is done via normal communication paths.
Unicore client program connects to Unicore gateway. This connects internally to the Network Job Supervisor service</td><td rowspan="3">Protocol uses fixed configurable TCP port. Communication and access is allowed with certificates only. So there is only low security impact.</td></tr>
<tr><td><b>Ports used</b></td><td>One TCP port
<i>Port is configurable</i>
Depending on location of the NJS an additional port may be needed to be opened</td></tr>
<tr><td><b>Protocol used</b></td><td>TCP</td></tr>
</table>

Name	Firewalls and high bandwidth, long distance networks					
Description	This use-case describes a setup that allows the creation of (optical) by-pass connections that span long distances which need to be connected via a firewall					
	Elements in communication path	**Software**		**Hardware**	**Network**	**Security Policy**
Severity		Low		High	middle	High
Occurrence		NA		performance	management	management
	Enterprise and public firewalls at both ends of a connection. Enterprise firewall both connects to the DMZ and to an optical by-pass connection.	**Own Software**	Yes and No GridFTP or any other datamover may be used – requirements are independent	Switching performance and buffer space is critical for the enterprise side of the firewall. Buffers should be able to contain the bandwidth/delay product of a long haul connection. Performance should be in the multi-Gb range.	Enterprise firewall may be involved in driving the request of a by-pass connections when detecting private address space ARP requests or handling application specific signals using some protocol	1. Requests from an application to access the optical by-pass should be authorized. The firewall should call out to obtain such authorizations or be provisioned with information that recognizes an access request. 2. Security policies should prevent hi-bandwidth / non-TCP transmission protocol conformant traffic to be leaked into the regular Internet.
		Ports used	Globus port range or others			
		Protocol used	TCP and UDP in various flavors			

<table>
<tr><td><b>Name</b></td><td colspan="6">Web Services Firewall Issues</td></tr>
<tr><td><b>Description</b></td><td colspan="6">Clients outside a network protected by a firewall must be able to refer to the Web Service End Point Reference (EPR)</td></tr>
<tr><td></td><td><b>Elements in communication path</b></td><td colspan="2"><b>Software</b></td><td><b>Hardware</b></td><td><b>Network</b></td><td><b>Security Policy</b></td></tr>
<tr><td><b>Severity</b></td><td></td><td colspan="2">High</td><td>NA</td><td>Low</td><td>High</td></tr>
<tr><td><b>Occurrence</b></td><td></td><td colspan="2">NA</td><td>NA</td><td>NA</td><td>Management</td></tr>
<tr><td rowspan="3"></td><td rowspan="3">The server's network is protected by a firewall and a SOAP-proxy firewall in parallel, which acts as a gateway between external clients and WS Application Server.
Any other kind of firewall may be located between the client and the server.</td><td><b>Own Soft-ware</b></td><td>External clients must know to refer to the SOAP-proxy in order to reach Web Service EPRs.
Internal EPRs must be translated to external EPRs, in order to be reached through the SOAP-proxy.</td><td rowspan="3">Web Services are running on hosts located in the internal network.</td><td rowspan="3">Firewalls in the communication path may not allow direct connections.</td><td rowspan="3">It is not possible to know how many Web Services are running on a single port.
No way to express a policy that informs client to extend the security context end-to-end when communicating through the SOAP-proxy.
The SOAP-proxy must have the same or higher level of trust when EPRs are communicated to external clients.</td></tr>
<tr><td><b>Ports used</b></td><td>SOAP over HTTP (port 80).
More than one Web Service may run on the same port.</td></tr>
<tr><td><b>Protocol used</b></td><td>TCP</td></tr>
<tr><td></td><td></td><td colspan="4">There is no standard mechanism to
• Augment an EPR with routing information
• Obtain an external EPR from an internal EPR
• Publish and discover external EPRs</td><td></td></tr>
</table>

12 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

13 Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

14 Full Copyright Notice

15 References

[GridFTP-1] Allock,W. (Editor), GFD-20: GridFTP: Protocol Extensions to FTP for the Grid, Open Grid Forum, April 2003

[GridFTP-2] Mandrichenko,I. (Editor), GFD-21: GridFTP Protocol Improvements, Open Grid Forum, July 2003

[GridFTP-3] Mandrichenko,I. (Editor), GFD-47: GridFTP v2 Protocol Description, Open Grid Forum, May 2005

[RFC 1631] Egevang,K., Francis,P., The IP Network Address Translator (NAT), RFC 1631, May 1994

[RFC 2663] Srisuresh,P., Holdrege,M., IP Network Address Translator (NAT) Terminology and Considerations, RFC 2663, August 1999

[RFC 3234] Carpenter,B. Brim,S., Middleboxes: Taxonomy and Issues, RFC 3234, February 2002

[RFC 3303] Srisuresh,P., Kuthan,J,, Rosenberg,J., Molitor,A., Rayhan,A., Middlebox communication architecture and framework, RFC 3303, August 2002