GWD-I
FI-RG

Thijs Metsch (Editor), German Aerospace Center
Leon Gommans, University of Amsterdam
Egon Grünter, Research Center Jülich
Ralph Niederberger, Research Center Jülich
Alan de Smet, University of Wisconsin
Gian Luca Volpato, RRZN LUH

February 27, 2008
Updated: February 27, 2008

Requirements on operating Grids in Firewalled Environments.

Status of This Document

This document provides information to the Grid community about firewall issues in grid environments. Distribution is unlimited.

Copyright Notice

Trademark

OGSA is a registered trademark and service mark of the Open Grid Forum.

Abstract

This document describes and evaluates approaches and solutions for firewall issues, such as application level gateways, host based firewalls, VPN style gateways etc., which are currently available to solve some of the problems Grid applications may face when dealing with firewalls located between the source and the destination of their communication paths.

Contents

Abstract 1
1. Introduction 2
1. Requirements from documenssst #1 2
1.1 General requirements 2
1.2 Requirements on hardware on the communication path 3
2.3 Requirements for the support of Grid Middleware solutions/protocols 3
2.4 Requirements on Data transfers and storage 3
2.5 Requirements on performance and configuration 3
2. Solutions 4
2.1 High speed firewalls 4
2.2 Load balancing firewalls 4
2.3 Dyna-Fire 6
2.4 Cooperative On-Demand Opening 6
2.5 Generic Connection Brokering 7
2.6 UDP Hole Punching 8
2.7 Application Level Gateway / Proxies 8
2.8 A framework for Token Based Firewalling in Hybrid GMPLS networks - Policy based Access Control requirements in GMPLS networks to support Grids 8
3. Matching requirements and solutions 8
4. Summary 8
2. Contributors 8
3. Glossary 8
4. Intellectual Property Statement 8
5. Disclaimer 8
6. Full Copyright Notice 8
7. References 8

1. Introduction

This document presents an overview of existing and upcoming solutions for firewall issues in grid environments. It summarizes the requirements on firewalls and tries to match them to appropriate sets of solutions. Requirements that can not be covered by any existing solution are analyzed further on in this document.

The Firewall Issues Research Group has published in August 2006 a first document [FI-RG-1] that collects and classifies the requirements of grid applications in terms of firewall functionalities. So far, the following categories have been identified:

1. Some applications need high-throughput data channels; firewalls must be able to cope with large amount of data traffic.
2. Some applications use only a set of well-known ports.
3. Some applications use a set of well-known ports and a set of detectable dynamic ports. Detectable means that through analysis of the traffic between the server and the client it is possible to determine which dynamic ports will be used by the applications.

Some applications use a set of well-known ports and a set of obscured (non detectable) dynamic ports. “Obscured” means that the analysis of the traffic between the server and the client does not provide enough information to determine which dynamic ports will be used by the applications. This may happen because the traffic is encrypted or because the necessary information is not exchanged at all between server and client. Because of the lack of knowledge, firewalls are not able to recognize which data streams (well-known port connections and dynamic port connections) belong together.

1. Requirements from document #1

The following requirements have been identified in the first document [FI-RG-1].
This short summary classifies the requirements into four groups. Every group is described by a table holding an identifier for the requirement, a short description, and a severity level of the issue (low, medium or high).

1.1 General requirements

ID	Requirement	Severity	Details
G010	Applications use a single port/socket connection for simple data transfers	Medium	Using a single (well-known) port allows firewall administrators to open only a small set of ports and only for a limited number of specified servers (minimum uncertainty).
G020	Grid resources need to be protected by firewalls	High	Grid resources have to be considered as any other asset, with the peculiarity of being reachable from outside. Moreover these resources are often well known by outsiders and are appealing for attacks, because of large storage capacity, extreme computing power and high bandwidth connectivity.
G030	The number of ports needed by an application must be detectable before, after, and during its execution	High	It is always recommended to use well formed program structures, so that at any time full control and knowledge of the status of the running tasks are available. Using only well-known ports allows best

			monitoring of communication streams.
G040	The network and the applications need to deal with security policies	High	Applications, that are not aware of security issues, should not be allowed to use the network. Though a netiquette is well known, there are always people which do not bother respecting these “hindering” rules.

1.2 Requirements on hardware on the communication path

ID	Requirement	Severity	Details
H010	Routers/Firewalls need to deal with high-bandwidth networks	High	Firewalls are usually located at ingress/egress points of organizations and they have to deal with any kind of traffic: both “single streamed” or aggregated.
H020	The number and type of firewalls along a routing path need to be known	Medium	Both the number and the kind of hardware located within a routing path need to be known or the application has to provide network relevant info via the path over which the communication is taking place..

2.3 Requirements for the support of Grid Middleware solutions/protocols

ID	Requirements	Severity	Details
M010	Support connection from an ephemeral (so unknown) port on a client to a grid service port on the server	High	To support Globus Toolkit (GT) or Unicore
M020	Support connection from a controllable ephemeral (so unknown) port on a client to a grid service port on the server	High	To support GT
M030	Support connection from a controllable ephemeral (so known) port on a client to a controllable ephemeral (so known) port on the server	Medium	To support GT
M040	Support the usage of internal and external service EPRs	Medium	
M050	Support the usage of ephemeral (so unknown) internal EPRs which are mapped to external EPRs	Medium	
M060	Support ports used by group collaboration systems	Medium	Audio and video ports needed by AccessGrid

2.4 Requirements on Data transfers and storage

ID	Requirements	Severity	Details
D010	Use a single socket connection for control channels	High	GridFTP control channel, GSIdCap for dCache, SRM protocol
D020	Support the usage of arbitrary numbers of sockets (which can vary in time)	High	GridFTP data channels, i.e. predefine or signal port usage

2.5 Requirements on performance and configuration

ID	Requirements	Severity	Details
P010	Support high performance network	High	Special applications require high

	links with bandwidth up to 10/40 Gb/s		speed data connections. Allow a) high speed (single stream), b) load balancing or c) firewall bypass
P020	Support dynamic configuration of firewalls by authorized users	High	Define an adequate protocol with authentication and authorization functionalities or use/misuse existing protocols.

2. Solutions

This section describes existing or upcoming solutions which try to solve the requirements listed above.

2.1 High speed firewalls

Most of the current high-speed firewalls are able to process up to 5 or 6 Gb/s of data traffic, because of the speed limit of their internal CPU and forwarding engines. Often special hardware is used to achieve such an high throughput. A market overview of high speed firewalls (as of August 2006) can be accessed at the D-Grid project web page [D-Grid].

Current routers could act as a packet filters as far as it fits the needs of the local security policy. In this case performance up to tens of Gb/s could be achieved. However only a limited number of routers is able to handle the extended access lists required to perform stateful inspection. If stateful inspection has to be done a packet filter solution is insufficient.

2.2 Load balancing firewalls

An alternative solution to achieve high rates of firewall throughput is the use of "load balancing" configurations. Multiple firewalls can be deployed for scalability and reliability, and simultaneously provide balancing of the traffic on inbound and outbound paths. There are different ways to realize this idea.

Multiple firewalls are combined into a firewall farm. One firewall is the firewall master and delegates traffic to the other firewall slaves. It may use a hashing algorithm based on information of the TCP/IP headers to allocate traffic to the slaves. The firewall master uses ICMP-Redirects to specify which firewall slave will deal with the incoming packet. The redirect information is sent to the routing instance in front of the incoming interface. This algorithm guarantees that routers and firewalls of different vendors interact well. In fact with this algorithm single streams are balanced rather than the load itself, i.e. a single stream of 10 Gb/s is transferred through one single firewall slave.

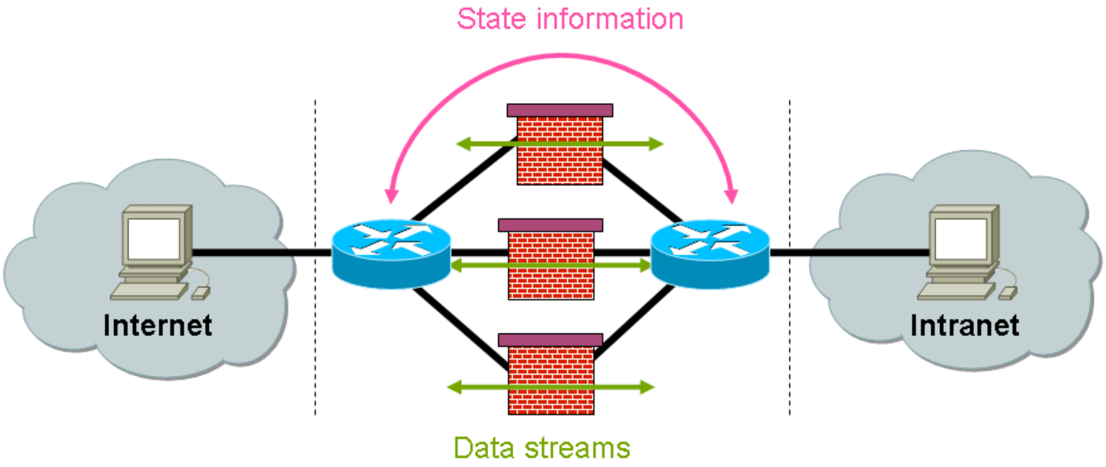

Figure 1: master/slave load balancing controlled via ICMP redirects

Another approach is based on the usage of a round robin algorithm. Each incoming packet is delegated to a different firewall. However, since this is real load balancing and streams are distributed across different firewall slaves, new problems arise. First of all an algorithm for packet routing has to be defined. Using ICMP-Redirects results in additional load and this leads to degradation of the network performance. The introduction of special load balancing devices in front and behind the firewall farm could solve this problem. These devices use routing tables and exchange routing information. The second problem is the choice between synchronous and asynchronous firewall implementations: synchronous firewalls share connection information among each other, so that data of a given connection can go through any firewall. Asynchronous firewalls do not share connection information and traffic must be revalidated each time it goes through a new firewall. Because of the huge amount of state information to be exchanged the round robin algorithm is currently not used in any of the firewalls available in the market.

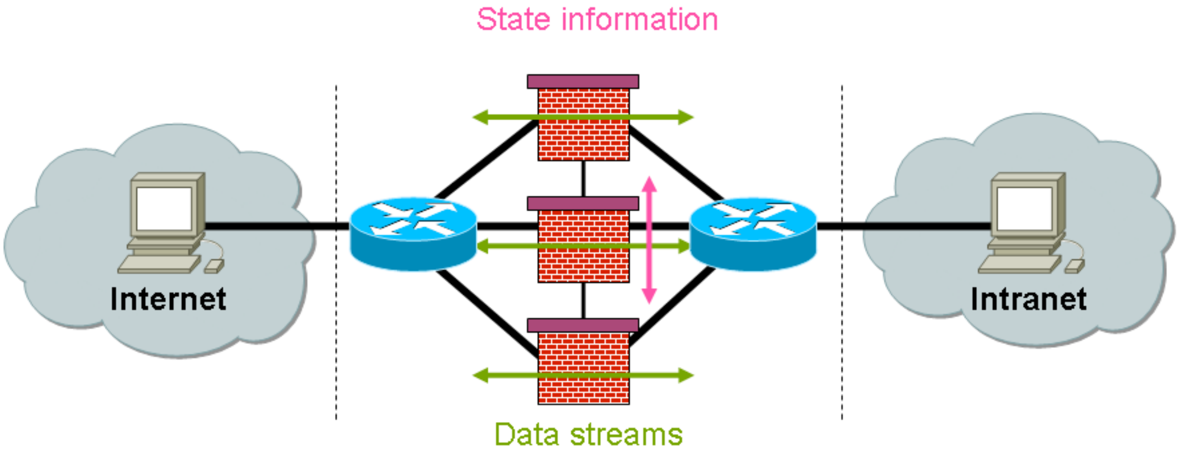

Figure 2: load balancing devices

The use of load balancing devices in front of and behind the firewall is a common approach. Although they do not currently use any round robin approach, they use configurable algorithms. Some vendors allow the definition of certain hash algorithms and load balancing based on session identifiers. Some devices provide also proper handling of protocols like.FTP. However it is not sufficient to use such devices as long as they do not fit the performance requirements.

Currently, load balancing is based on hashing algorithms using address and port information from the protocol headers. Though load balancing systems allow the distribution of different streams to different firewalls, they do not implement load balancing of one single high-speed data connection.

2.3 Dyna-Fire

Dyna-Fire is an extension of the Netfilter iptables software capable of configuring new filtering rules in response to valid requests from users' applications. Its main features are:

- The use of Port Knocking as signaling protocol.
- The storage in a central database of all users and resources information.
- The integration with the Globus Gatekeeper.

Port Knocking signaling consists in a client-server communication method in which information is encoded in the form of connection attempts to closed ports. The server initially presents no open ports to public networks and monitors all connection attempts. A client initiates a connection by sending TCP packets carrying SYN flag to a specific sequence of ports. When the server detects a valid knock sequence it triggers a server-side process, usually the opening of the requested port. The knock sequence may contain additional information such as the IP address of the client host, the requested resource (application name or port number), the opening time of the port, and a user identifier.

The Dyna-Fire daemon is integrated into the existing network firewall (iptables). In this way it can monitor connection attempts from external entities. When it detects a valid knock sequence it tries to validate it. If all verification criteria are satisfied Dyna-Fire adds a new filtering rule into the iptables firewall. This rule allows access to the destination host and port number only from the IP address that sent the knock sequence and only for a length of time corresponding to the validity of the user proxy certificate.

Dyna-Fire has been developed at the Center for Computational Research, University at Buffalo, New York. Though the current implementation is available only for iptables firewalls, Dyna-Fire could be used for any kind of commercial firewalls by integrating additional code into the firewall software stack or by analyzing syslog output. Whereas the concept seems promising a closer look shows a large overhead in packet flows and confirmation data. Getting one additional open port in the firewall requires a lot of signaling (port knocking) packets to be sent to provide the desired port, the allowed IP addresses, and user info. [Dyna-Fire]

2.4 Cooperative On-Demand Opening

CODO is an extension of the Netfilter iptables software capable of configuring new filtering rules in response to valid requests from user applications. Its main distinctive features are:

- The use of SSL for the signaling channel.
- The use of a memory table to register all applications authorized to receive connections from the outside.
- The ability to control both incoming and outgoing connections.

The existing network firewall is integrated with a CODO Firewall Agent. The firewall initially presents only three open ports. The CODO agent uses these ports to listen for signaling messages: one port is used by internal applications to register themselves as available destination entities; another port is used by external applications to request connections to internal destination entities; the third port is used by internal applications to request connections to external destination entities.

All signaling messages are exchanged over an SSL channel enforcing mutual authentication with X.509 digital certificates. The CODO agent defines in its configuration parameters a list of trusted Certification Authorities. Requests coming from external applications are allowed only if the destination entities are already registered in the memory table. If all controls are successfully passed the CODO agent adds a new filtering rule into the iptables firewall. This rule allows access to the destination host and port number only from the IP address and port number indicated in the connection request. [CODO]

2.5 Generic Connection Brokering

Generic Connection Brokering enables firewall traversal by decoupling the direction in which connections are established between clients and servers. A brokering system arranges which party should initiate the communication, based on the network configuration of each party.

In the traditional Berkeley socket system a connection is always established from the client to the server, however when the server is placed behind a firewall it usually cannot receive incoming connections. GCB implements an intermediate layer between the application software and the Berkeley sockets and it reverses the direction of the connection without application's notice. GCB broker decides, based on the network situation of the client and the server, who should actively connect and arranges accordingly. If neither can connect to the other, because for example they are both inside a protected network, both parties connect to the GCB broker and it relays packets between them.

GCB requires that applications be linked with GCB libraries. Applications linked against GCB should have no change in their behavior until the GCB functionality is specifically enabled. GCB aims to provide semantics as close to Berkeley sockets as possible, but some applications will require minor modifications to take advantage of GCB's functionality. All connections using GCB require contacting the broker, either to negotiate a connection or to use the provided relay (client can still directly contact the server, but will not get the firewall traversing capability of GCB). To support GCB the function getsockname() must return the IP address of the broker, not the server. The server is expected to transmit this address to potential clients using existing advertising functionality [GCB-1].

As described above it is possible for a GCB-enabled server hidden behind a firewall to be contacted by a non-GCB enabled client. In this case all communications are relayed by the broker. GCB requires one or more "brokers" to handle negotiating and relaying connections. All machines involved need to be able to make outgoing connections to the broker.

GCB is in use by several groups using Condor's "glide-in" functionality to create dynamic Condor pools of hundreds of machines on top of existing grids.

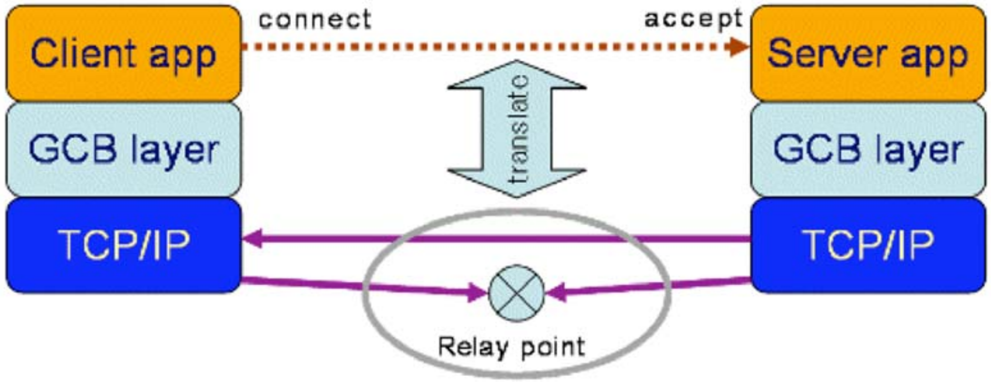

Figure : GCB architecture (© http://www.cs.wisc.edu/condor/gcb/basic.html)

2.6 UDP Hole Punching

UDP hole punching is a method for establishing bidirectional UDP communications between hosts protected by firewalls. It can be used to establish communications across firewalls, by means of simulating UDP connections. As a prerequisite to use UDP hole punching firewalls must allow outbound TCP and UDP connections.

A relay server, which controls the establishment of a secure communication, has to be installed. The server is a central part of this concept. Each client connects to this server using a persistent TCP channel. The relay server conveys the public IP address of each client to the other so communication between the two can be initiated.

A connection is established through several steps (see Figure 4). The initiator (client A) sends a TCP segment to the relay server containing the information that it wants to talk to client B using a specific dynamic UDP port. The server notifies client B about client A's IP address and that it expects a UDP connection on this specific port. Client B sends his dynamically generated UDP port number also to the relay server and simultaneously sends a UDP datagram from this port to the earlier predefined port of client A.

Client B's local firewall forwards the UDP datagram, stores a connection entry and the dynamic access rule which allows responses to traverse the firewall into its internal database. In this phase Client A's local firewall rejects the packet, but this does not matter at all. The relay server informs client A via the existing TCP connection to A that client B is accessible on the predefined UDP port.

Client A now sends a UDP datagram to B. Client A's local firewall creates also a dynamic entry in its database. However, the dynamic entry in B's local firewall is still active and valid, so that the UDP datagram from A to B passes the firewall. The communication channel is now established, despite the static rulesets of each firewall would normally deny inbound connections.

The concept of UDP hole punching can easily be adapted to work also in Grid environments [GUDPH]. The relay server is extended to an authentication and authorization service that is located inside client B's domain on a special server or directly onto host B. It listens on a well defined TCP port and waits for user connections. At connect time, the remote user and the server authenticate each other using X.509 certificates. If the TCP connection between the participants has been established UDP data transfers can take place as described in the UDP hole punching concept above.

The advantage of an AAA service is obvious. The relay server and client B can be installed on the same host, without using an outside relay server that is not administrated by one of the involved organizations. Mutual authentication between client and server is necessary only at connect time. Once the TCP connection is established data transfers are allowed to start. Furthermore there are no problems resulting from NAT. Any server accessible from the public network is exempted from the NAT algorithm and the server only sees the public IP address of the client.

The security impact of Grid UDP hole punching depends directly on the secure implementation of the AAA service. If authentication and authorization of remote users/processes has been programmed with standard and up-to-date security tools, then the dynamic firewall rules can be created because access rights to the service have been assigned by the administrator of the service.

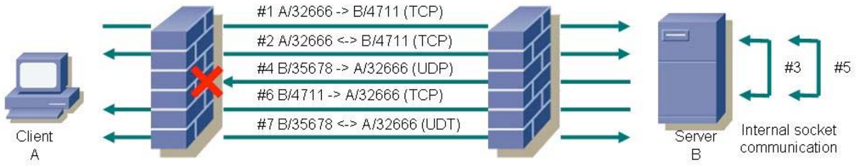
#1 A/32666 -> B/4711 (TCP)
#2 A/32666 <-> B/4711 (TCP)
#4 B/35678 -> A/32666 (UDP)
#6 B/4711 -> A/32666 (TCP)
#7 B/35678 <-> A/32666 (UDT)
Client A
Server B
#3
#5
Internal socket communication

Figure 4 UDP hole punching in Grid environments

Often it is useful to transfer data using multiple parallel connections. The client has to indicate how many connections should be used in parallel. It allocates the sockets and sends the port numbers to the server. The server reads the information and tries to initiate the maximum number of UDP connections available by sending small UDP datagrams to the client. This may depend on the system load. The server sends the local UDP port numbers to the client and the client starts the data transfers. Starting two TCP connections, one from A to B and one from B to A, is not possible, because the firewall recognizes the mismatching TCP sequence numbers and deletes the dynamic access entries.

UDP data traffic can be easily encrypted using a simple symmetric algorithm. Because the TCP connection between client and server is secured via X.509 certificates the exchange of a shared secret could be done via this channel.

Data transfers within Grid applications should be fast and reliable. Because UDP has no three-way-handshake and no acknowledgement it is faster than TCP, but it is not reliable. Each UDP datagram is an instance of its own and the application has to make sure that all the data has arrived. In fact this means that Grid applications have to be modified if they want to use UDP as the underlying transport layer.. This can be accomplished by the UDP-based Data Transfer Protocol (UDT) [UDT06].

UDT uses UDP as transport protocol but it guarantees reliability through an upper layer. Of course Grid applications have to be modified also, but the effort is very small because an API using UDT can be provided. Only interfaces of subroutines allocating sockets have to be changed.

UDT is available as open source and distributed under the LGPL. It is designed and implemented by the National Center for Data Mining at the University of Illinois at Chicago. A first internet draft has been released in August 2004.

2.7 Gateways / Proxies

To securely access Grid resources in networks protected by firewalls, a security proxy for Grid Services can be developed. The main aspect of this concept is a security gateway which performs content based checks on incoming Grid requests. This is an application level gateway (ALG) and it checks SOAP messages of Grid Service requests and decides on the application level (OSI level 7) whether the message should pass the gateway or be blocked. In combination with packet filtering, provided by usual firewall solutions, and encrypted data transfer methods, this allows a shared secured use of Grid resources, separated by security gateways. This can be accomplished without major changes to the respective Grid middleware (as Globus or UNICORE) and without increasing security risks to an unacceptable level (e.~g., by opening network ports).

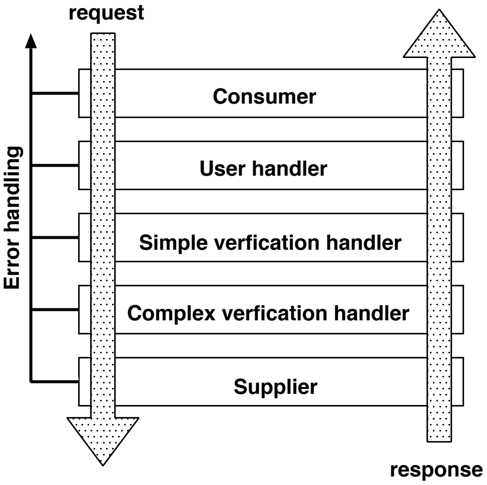

Figure 5 Message passing within a gateway

The figure above describes the path a message takes within an gateway. First of all the incoming requests have to be accepted by the gateway. The consumer is the instance receiving the request. The data arriving here is most likely SSL encrypted. The message is transferred inside the gateway system without encryption. In the consumer the incoming decrypted SOAP messages are parsed.

Next the certificates provided with the request need to be verified. Handlers are provided to accomplish this task. A very strict user control can be implemented at this point. It can be determined which user is allowed to invoke which service. If the user is allowed to invoke a service, the message is passed on to the next step. Otherwise the request is rejected and an error message is returned to the client.

The first step within the verification process is the verification of the SOAP message (done in the verification handlers).

Within the more complex verification it is possible to check and alter messages. For example executables could be scanned for viruses. Messages can be modified in place, e.g. a path to executables could be replaced by an alternative path unknown to the user. Even firewalls can be open to support GridFTP transfers.

The last step of the verification process is the supplier. It sends the verified messages to the final destination.

Within a gateway information about the user and about the network resourced used can be added to the SOAP message.

The gateway can be used to address different security concepts:

- Using an ALG as Web Service Proxy - Authentication of users with the help of GSI (No knowledge about what is actually going on).
- Validation of all actions taken by users Knowledge about all operations can be gained (Accounting, Logging) (Increases latency).
- Usage as a “firewall opener” - Support for non-blocking buffered I/O (RFT/GridFTP) (Can become complicated).

2.8 A framework for Token Based Firewalling in Hybrid GMPLS networks - Policy based Access Control requirements in GMPLS networks to support Grids

Generalized Multi Protocol Label Switching (GMPLS) is becoming increasingly popular as a control plane solution for optical networks working in parallel with a common routed IP network. GMPLS deploys a separate data forwarding plane and control plane. The control plane uses the ReSerVation Protocol (RSVP), modified to allow Traffic Engineering (TE) as signaling protocol. “Token Based Firewalling in Hybrid GMPLS networks” describes an innovation that enhances the secure use of GMPLS based networks by creating a firewall function that can sit between a routed (connection-less) packet based network and the switched (connection-oriented) GMPLS network, which needs protection against unauthorized usage. Tokens will be used to mark the authorization of certain IP streams to make use of resources inside a GMPLS network. The firewall will route non-authorized IP streams to a transit network.

GMPLS RFC3471 is a generalization of the MPLS architecture RFC3031, which is intended to create connections across packet- or cell (ATM) based networks. In MPLS, packets or cells are forwarded using labels carried inside the data. Such an approach requires a minimum amount of processing during the forwarding process as a label already identifies a path, which is pre-established by a separate control plane using a signaling protocol called RSVP RFC2205. GMPLS removes the requirement for switching devices to be able to recognize packets or cells and consequently the requirement to forward data based on label information. As such, GMPLS adds capabilities to forward data based on for example time-slot numbers, wavelengths, physical ports or Virtual LAN number instead. In order to support signaling for additional classes of switching (based on e.g. Time, Wavelengths and Ports) inside the GMPLS control plane, the RSVP protocol has been enhanced with Traffic Engineering extensions (RSVP-TE) as defined in RFC3473.

The basic security mechanisms of MPLS using RSVP signaling are defined in RFC2747. The defined mechanisms provide signal message integrity and node authentication. Apart from an additional end-to-end security requirement for notification messages, the GMPLS standard itself does not impose additional security requirements. Furthermore, the security considerations of RSVP-TE in RFC3473 state that the control plane of a GMPLS network must not accept any message from nodes that are not known to the recipient to be authorized to make such requests. GMPLS was originally designed to serve on-demand requests. Therefore, the information kept at each end to maintain a security association, does for example not identify a particular reservation. The security association kept between two corresponding nodes using RSVP maintains only information about the keys lifetime, the information identifying the sending station and sequence number information. Such state may therefore be insufficient to support a more complex set of access control requirements, which Grid middleware systems may demand from a GMPLS network.

When considering the concept of a GMPLS domain, RFC4726 recognizes that the ingress point of a GMPLS domain is a natural point for policy control. In order to assist one or more Grid-(meta-)scheduler to ensure that assumptions of network availability are accurate, it helps if the ingress point can enforce access based on a secured reservation-id. As multiple scheduling agents may reserve network resources, the concept of accepting a ticket or secure token at the ingress point of a network has proven to be pragmatic. Many examples of such a system can be found in transportation networks such as airline-, train- and bus networks. In our research case, the RSVP protocol allows ticket or secure token information to be contained in a POLICY_DATA object as described in RFC2750. In this context, we define a secure token as a signed list of attributes, where at least one attribute identifies a particular service instance. In order to recognize the authenticity and integrity of the token, key material must be shared between the authority issuing the right to access the service instance and the network ingress point that enforces the access. The authority may be a complex set of middleware functions, which includes a scheduler. As such the GMPLS Network ingress point can be considered as a firewall, admitting only an Authorized Label Switched Path (ALSP). Figure 5 below shows some basic network elements around the concept of a GMPLS firewall.

The GMPLS firewall sits in between a network that connects a GMPLS client. In effect the GMPLS firewall acts as a proxy for the GMPLS Ingress LSR to which it would normally connect. The GMPLS firewall is connected to the GMPLS Ingress LSR and to the Routed Internet, offering transit (i.e. connectivity to the entire Internet). Both the Firewall and GMPLS client are provisioned with the proper key material, such that the content of the POLICY_DATA object can be signed by the client and recognized by the GMPLS firewall. The content of the POLICY_DATA object is dependant on the application, but for scalability reasons it should be minimal. If the content of the POLICY_DATA object, acting as a secure token, only refers to some pre-provisioned behaviour, that is programmed into the firewall, it has the advantage that the GMPLS client does not have to be aware of- or required to handle the associated attributes. The pre-provisioned behavior could identify future time-slots, source/destination constraints, route and bandwidth constraints, etc. As a minimum the POLICY_DATA object should carry some ID and signature.

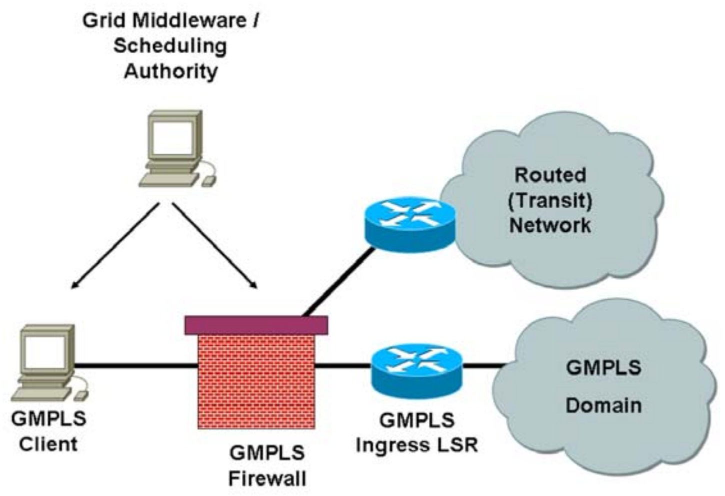

Figure 6 GMPLS

The above concept is intended to act as a framework describing an approach to use the RFC2750 POLICY_DATA object as secure token. Further research needs to be performed to identify what attributes should be carried inside the POLICY_DATA object, how key material should be distributed and what policies should govern the issuing of the keys, in particular in multi-domain GMPLS network scenario’s.

3. Matching requirements and solutions

This section provides an overview of which firewall requirements are addressed by the solutions previously presented. In the summary table several fields are left blank on purpose when they are out of scope for the examined solution. The solutions currently available focus on the following categories of problems:

- Use of high-bandwidth network infrastructure
- Dynamic configuration of firewalls
- Enforcement of policies for secure data transfers

Name of Solution / ID of requirement	G010	G020	G030	G040	H010	H020	M010	M020	M030	M040	M050	M060	D010	D020	P010	P020
High speed FWs	🟩	🟩	N/A	🟩	🟩	🟥	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	🟩	N/A
Load Balancing FWs	🟩	🟩	N/A	🟩	🟩	🟥	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	🟩	N/A
Dyna-Fire	🟩	🟩	🟩	🟩	N/A	🟥	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟥	N/A	🟩
Cooperative On_Demand Opening	🟩	🟩	🟩	🟩	N/A	🟥	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟥	N/A	🟩
Generic Connection Brokering	🟩	🟩	🟩	🟩	N/A	🟩	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟥	N/A	🟥
UDP hole Punching	🟩	🟩	🟩	🟩	N/A	🟥	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟩	N/A	🟩
Application Level Gateway / Proxies	🟩	🟩	🟩	🟩	N/A	🟥	🟩	🟩	🟩	🟩	🟩	🟥	🟩	🟩	N/A	🟩
Token[RRN1] Based FWing in Hybird GMPLS Networks	🟩	🟩	🟩	🟩	N/A	🟥	🟩	🟩	🟩	🟥	🟥	🟩	🟩	🟩	N/A	🟩

This table clearly shows that requirements H020, M040, M050, and D020 are currently not yet supported by most of the solutions. These requirements are:

ID	Requirements	Severity	Details
H020	The number and type of firewalls along a routing path need to be known	Medium	Both the number and the kind of hardware located within a routing path need to be known or the path over which the communication is taking place.
M040	Support the usage of internal and external service EPR	Medium	
M050	Support the usage of ephemeral (so unknown) internal EPRs which are mapped to external EPRs	Medium	
D020	Support the usage of arbitrary numbers of sockets (which can vary in time)	High	GridFTP data channels, i.e. predefine or signal port usage

The requirement that presents the highest severity level is D020. The other requirements come along with the severity medium. This leads to the conclusion that data transfers still suffers from major drawbacks when operated in firewalled environments. The mapping of internal Endpoint References to external ones also constitutes a problem, especially when using ephemeral (known or unknown) ports. Knowledge of the number and type of firewalls (and other network devices) on the routing path is very convenient in many occasions, e.g. if you need to debug a grid communication problem.

New and upcoming solutions should take into consideration the need of addressing these requirements.

4. Summary

The previous chapter described to what extent Grid requirements for Firewalls are satisfied by the currently existing solutions. The different solutions might be used on a case-by-case basis, having previously analyzed the requirements of the security policy of the individual organizations. None of them fit all needs, but most of them have their right to exist. What's missing up to now is a general solution that fits most of the needs, is simple to implement, highly responsive (throughput), globally applicable (standard well known as well as dynamic connections), secure, highly recognized (commonly known and implemented) and last but not least standardized.

While this document gives mostly an overview of the requirements and existing solutions some issue came up during evaluating the solutions:

- Grid Middleware resources and network resources (like firewalls, routers and load balancers) are still two completely seperated layers in Grid environments.
- Knowledge of the complete communication path would help abstracting network resources in a Grid.
- Support for this kind of network information is not supported by the middlewares currently.

Grid Middleware and Network resources management are still complete seperated working fields. To resolve many of the issue described in our first document the coupling of these two wokring fields becomes necessary. Goal is to make Grid middleware solutions aware of the networking resources and vis versa. Therefore protocols and services need to be defined which allow:

- The grid middleware to pass on network information.
- The grid middleware to alter network configurations (like dynamically open firewalls).
- The new, to be developed, solution to communicate with network resources.

This working group plans to investigate on this issue in the future. The objective is to design, implement, and validate such a solution for Grid environments.

2. Contributors

Name	Institution
Alan De Smet	University of Wisconsin adesmet@cs.wisc.edu
Leon Gommans	University of Amsterdam lgommans@science.uva.nl
Egon Grünter	Research Center Jülich e.gruenter@fz-juelich.de

Thijs Metsch	German Aerospace Center - DLR e.V. thijs.metsch@dlr.de
Ralph Niederberger	Research Center Jülich r.niederberger@fz-juelich.de
Gian Luca Volpato	RRZN - Leibniz Universitaet Hannover volpato@rrzn.uni-hannover.de

3. Glossary

AccessGrid	AccesGrid is a collection of resources and technologies for audio and video collaboration between workgroups working apart.
ALG	Application Level Gateway, see chapter 3.3
Condor	The goal of the Condor® Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Guided by both the technological and sociological challenges of such a computing environment, the Condor Team has been building software tools that enable scientists and engineers to increase their computing throughput. (© http://www.cs.wisc.edu/condor/)
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.
dCache	The goal of this project is to provide a system for storing and retrieving huge amounts of data, distributed among a large number of heterogenous server nodes, under a single virtual filesystem tree with a variety of standard access methods. 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. (© http://www.dcache.org)
D-Grid	The German Grid Initiative is a joint research initiative between German research and industry funded by the Federal Ministry of Education and Research (BMBF) to develop a distributed, integrated resource platform for high-performance computing and related services to enable the processing of large amounts of scientific dta and information.
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.
Globus Alliance	The Globus Alliance is an international collaboration that conducts research and development to create fundamental Grid technologies. The Grid lets

	people share computing power, databases, and other on-line tools securely across corporate, institutional, and geographic boundaries without sacrificing local autonomy. (© http://www.globus.org/alliance/)
Globus Toolkit	The Globus Toolkit is an open source software toolkit used for building Grid systems and applications. 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. (© http://www.globus.org/)
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.
IPSec	IP Security, 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).
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.
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
Unicore	UNICORE (Uniform Interface to Computing Resources) offers a ready-to-run Grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet. (© http://www.unicore.eu)
WSRF	The WS-Resource Framework (WSRF) is a set of six Web services specifications that define what is termed the *WS-Resource approach* to modeling and managing state in a Web services context. To date, drafts of three of these specifications have been released, along with an architecture document that motivates and describes the WS-Resource approach to modeling stateful resources with Web services. To be released soon are the other specifications, an overview document describing the relationship among the different specifications, and a document that compares the WS-Resource Framework with the Open Grid Services Infrastructure. (© http://www.globus.org/wsrf/faq.php#wsrf1)

4. 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.

5. 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.

6. Full Copyright Notice

7. References

[CODO]	S.Son, B.Allcock, M.Livny, CODO: Firewall Traversal by Cooperative On-Demand Opening, 14th IEEE Symposium on High Performance Distributed Computing, (HPDC14), Research Triangle Park, July 2005 http://www.cs.wisc.edu/ sschang/papers/CODO-hpdc.pdf
[D-Grid].	M.Meier, E.Gruenter, R.Niederberger, Th.Eickermann, Überblick zur Marktrecherche – Hochgeschwindigkeitsfirewalls, D-Grid Integrationsbericht Fachgebiet 3-5, Forschungszentrum Jülich, August 2006, https://bscw.zam.kfa-juelich.de/bscw/bscw.cgi/d127712/ueberblick_hochgeschwindigkeitfirewalls.pdf
[Dyna-Fire]	M.L.Green, S.M.Gallo, R.Miller, Grid-enabled virtual organization based dynamic firewall, Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing, ISBN 0-7695-2256-4, pp. 208-216, Nov. 2004
[FI-RG-1]	R. Niederberger, W. Allcock, L. Gommans, E. Grünter, T. Metsch, I. Monga, G. L. Volpato, C. Grimm Firewall Issues Overview. Open Grid Forum, Lemont, Illinois, U.S.A., GWD-I.083, August 2006. http://www.ogf.org/documents/GFD.83.pdf
[GCB-1]	GCB: Generic Connection Brokering, http://www.cs.wisc.edu/condor/gcb/

[GUDPH]	E.Gruenter, M.Meier, R.Niederberger, F.Petri,, Dynamic Configuration of Firewalls - Using UDP Hole Punching, Oct. 2006, http://www.d-grid.de/fileadmin/user_upload/documents/DGI-FG3-5/Dynamic_Configuration_of_Firewalls.pdf
[UDT06]	Y.Gu, R.L.Grossmann, UDT – UDP based Data Transfer: Breaking the Data Transfer Bottleneck:, 2007, http://udt.sourgeforge.net