

A Report on Experiences Operating the Globus Toolkit through a Firewall

Version 1, September 2001.

Mark Baker, Hong Ong, Garry Smith,
Distributed Systems Group,
University of Portsmouth,
Portsmouth, UK

Abstract

This report details experiences encountered whilst investigating the use of the Globus Toolkit through a firewall. The investigation centered on the use of a single firewall with a number of different configurations. The primary approach involved opening selected ports on the firewall, used by Globus services, and ‘pinning open’ other unprivileged ports to allow subsequent communications. The second approach was based on a port-tunneling paradigm, implemented using the Nexus Proxy extension to Globus.

1 Introduction

Firewalls are an important part of many organizations’ network security infrastructure. However, in order to participate in a Grid, an organization’s computational resources must be available to authorized grid users external to that organization, typically via the Internet.

This report investigates the use of the Globus Toolkit [Globus01] through a firewall. In particular it provides practical information that an organization might use to configure their firewall for use with the toolkit. Firstly this report presents the “port pinning” approach favoured by the Globus team. This involves opening specific firewall IP ports to allow Globus traffic to pass through freely. Two firewall configurations are considered and their effect on the successful operation of Globus highlighted. Secondly, the Nexus Proxy [nexus-proxy01], port tunneling extension to Globus was investigated. The aim of this software is to reduce the number of open firewall ports required, by multiplexing and tunneling all Globus communications through a specified port.

Section 2 of this report presents Globus network information required for opening firewall ports. Section 3 provides details the local test environment; including high-level details of firewall configurations. Section 4 provides brief details of firewall rule testing, while section 5 presents the firewall tests in detail introducing the Globus commands used and the resulting TCP connection requests applied to the firewall. Section 6 briefly discusses port range issues. Section 7 describes firewall port configurations, section 8 presents issues highlighted during Globus testing and section 9 provides a resolution to one of those issues. Section 10 concludes the port pinning investigation with a short summary. The Nexus Proxy is examined in section 11 and

test results presented. Finally, section 12 concludes the report with a summary containing recommendations and mentions some outstanding issues.

2 Globus Port Information

Table 1 presents the information Globus have published [globus-firewall01] detailing network traffic for the Globus Toolkit. In addition, it should be noted that all TCP connections have a return connection on a port above 1024. This information is used as the basis to configure firewall ports under the ‘port-opening approach’. The Globus team report that it is possible to restrict the return connection port numbers by setting the environment variable `GLOBUS_TCP_PORT_RANGE <min,max>`, causing the Globus libraries to create listeners with ports within the specified range. This mechanism removes the requirement to leave all unprivileged firewall ports open.

Globus Service	Globus Version	Networks Ports	Traffic direction
Gatekeeper	1.1.3	2119/TCP	To hosts executing the service
Grid Resource Information Service	1.1.3	2135/TCP 2135/UDP	
Grid Information Index Service	1.1.3	Site defined i.e. port 2136	
GSI SSH	All	22/TCP	
GridFtp	All	2811/TCP (control) >1024/TCP data return channel	
MyProxy	All	7512/TCP	From myproxy -init or myproxy -get-delegation to the MyProxy server

Table 1: Globus Network Traffic

The Globus team goes further to report an unresolved issue with firewalls that perform network address translation. In particular the problem lies with the name selected for use within a host or gatekeeper certificate:

“Clients expect to see the same name in the certificate that they get from doing a reverse lookup on the IP address they are connecting to. [...] the IP address they will be connecting to will most likely be that of the firewall instead of the actual host so they will expect to see the firewall’s name in the certificate.” [globus-firewall01]

Requests from external users would succeed if the firewall’s name were in the certificate, but then internal users would receive mutual authentication errors because they expect to see the actual hostname [globus-firewall01].

In view of this, for all tests conducted in this report, no Internet Protocol (IP) address masquerading was performed. This approach was considered appropriate because it is anticipated that most sites using the Globus Toolkit (such as universities, research laboratories, and corporations) will have their own assigned range of legal Internet addresses, even though a firewall is in place.

3 Test Environment

3.1 Platforms

Figure 1 depicts the test environment. The firewall interface *eth1* is designated as the external interface.

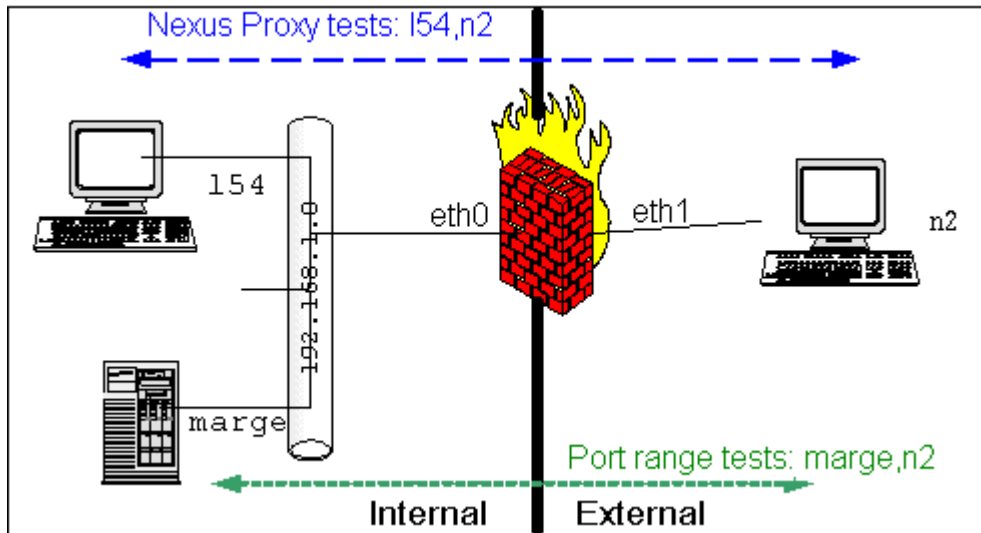


Figure 1: Test environment network configuration

Hostname	Platform	Interface(s): address
n2.dsg-cluster.csm.port.ac.uk	Redhat Linux 6.2, Kernel 2.2.14-5.0 i586 Globus1.1.3	eth0: 192.168.100.2
154.csm.port.ac.uk	Redhat Linux 6.2, Kernel 2.2.14-5.0 i586 Globus1.1.3	eth0: 192.168.1.4
marge.csm.port.ac.uk	Red Hat Linux 6.2, Kernel 2.2.17-33.SMP i686 Globus1.1.4	eth1: 192.168.1.1
firewall.csm.port.ac.uk	SuSE Linux 7.2, Kernel 2.4.4 i586 ipchains 1.3.9,17-Mar-1999	eth0: 192.168.1.100 eth1: 192.168.100.100 (eth1: external firewall interface)

Table 2: Test environment platform information

3.2 Test configurations

Primarily two types of test were carried out between:

1. A server (n2) inside the firewall, with a client (marge) outside.
2. A client (n2) inside the firewall with a server (marge) outside.

These tests were applied to two different firewall configurations:

- Config 1: Globus service consumer only.
- Config 2: Globus service consumer and provider.

Configuration 1 might be used at sites where users wish to interact with the Grid purely in a client role. For example, currently the Globus Toolkit has not been released for the Windows platform, but users of Windows based systems can utilise the Java Commodity Grid (CoG) kit to implement Grid client applications. These applications can then be used through the firewall to connect to Grid servers. In this configuration, attempts to connect to Grid services *within* the firewall fail, because the necessary ports required to access and use Globus from outside are closed.

3.2.1 Firewall rules

The firewall implementation, `ipchains` [ipchains00], selected for this project uses the rule targets outlined in Table 3. In addition, rule syntax is presented in the appendix. The firewall rules used during testing are presented later in this report. Even though the configuration files have been heavily commented with explanations for each rule, it is perhaps useful to have some background information to the firewall configuration.

Rule Target	Description
ACCEPT:	Allows the packet through
REJECT:	Drops the packet, but (if it is not an ICMP packet) generates an ICMP reply to the source to tell it that the destination was unreachable
DENY:	Drops the packet as if it had never been received
RETURN:	Identical to falling off the end of a chain immediately
REDIRECT:	Redirect (UDP or TCP) to another port: valid only for INPUT chain

Table 3: `ipchains` Rule Targets

For each network interface default queues exist (user defined queues may also be used); INPUT, OUTPUT and FORWARD. Packet filtering is achieved by appending rules to queues. Policies denote the action that occurs if a packet reaches the end of a default queue without matching any rule within that queue. In the firewall configurations to come, one should note that the policy for the input queue is always set to DENY. In other words, an incoming packet must be explicitly matched to a rule (i.e. a specified port and protocol) in order to pass through the firewall.

The first firewall configuration to be presented (see section 7.1) has the following rule architecture:

- DENY ALL,
- NO IP MASQUERADING,
- ALLOW ALL packets from the internal network out,
- ALLOW ALL replies (only) in,
- ALLOW connection requests on a restricted range of monitored ports,
- Unsolicited packets are blocked.

In addition, the second firewall configuration allows connection requests to the Globus ports listed in Table 1 (see section 7.2).

4 Initial Firewall Tests

4.1 IPChains rule matching

Before submitting the firewall configurations to real Globus network traffic, `ipchains` was used in a self-diagnostic manner to report how the firewall would react to data packets of a given type and to a given destination.

The following syntax provides such a facility:

```
Ipchains -C <chain> -s <source addr> <port> -d <destination addr>
<port> -p <protocol> -I <interface>
```

For example, `ipchains -C input -s 0/0 80 -d 0/0 80 -p TCP -I eth1`

This command checks the specified packet against the firewall rule base: Inbound TCP packet on `eth1` from the destination port 80 on any source host to port 80 on any destination host.

5 Globus Firewall Tests

The Globus test strategy for each firewall configuration comprised of the following areas:

- Submit a simple job to a chosen compute node:
 - a. With and without file staging.
- Batch job handling:
 - b. Submit the job,
 - c. Check the job status,
 - d. Retrieve the results.
- Search for compute node information in a GIIS/GRIS.
- Perform a remote file copy.

The actual commands used are presented next.

5.1 Globus commands used

These commands reflect those present in the Globus Quick Start Guide [globus-qck-gde01] and provide a suitable range to test Globus interaction through a firewall. The same set of commands are consistently used throughout this report.

ID	Command
2	<code>globus-job-run <hostname> /bin/echo Hello World</code>
3	<code>globus-job-run <hostname> -stage hello.ksh</code>
4	<code>globus-job-submit <hostname> /bin/hostname</code>
5	<code>globus-job-status <url></code>
6	<code>globus-job-get-output <url></code>

7	grid-info-search -h <hostname> -p <port> "(objectclass=GlobusComputeResource)" dn ostype
8	globus-rcp <hostname>:/tmp foo /tmp/bar
9	9a: execute globus-gass-server& on target machine (url is returned) 9b: On client: globus-url-copy <fromURL> <toURL>

Table 4: Globus Commands used for Firewall Tests

Commands 2 and 3 are interactive commands. `globus-job-run` is a wrapper script that calls `globusrun` with arguments indicating the command should be executed synchronously (i.e. the client command returns the results directly to the caller). In command 2, `hello.ksh` is an executable script that is staged over to `<hostname>`, executed and the results returned to the caller. The staged script is then removed from `<hostname>`. Commands 4, 5 and 6 are batch job commands to allow job submission, status monitoring, and the collection of results. A reference url for the submitted job is returned to the caller when command 5 returns. Command 7 is a synchronous command that performs a query on the specified information resource for Grid infrastructure data, i.e. retrieving the names and operating system types of all compute nodes known to the information source. Commands 8 and 9 both allow remote files to be copied, however, command 9 requires that the user `bg` into the remote Grid node first and start a `globus-gass-server` process to provide access to the remote node's secondary storage.

5.1.1 Connection Requests

In order to ensure that assumptions about Globus network traffic were correct, the firewall was first placed in a passive, logging mode, between two Globus-enabled machines. By monitoring incoming traffic on both firewall interfaces and using an ACCEPT-ALL policy, the TCP connection requests for each command were observed. See Table 5 for an example of the network traffic observed. In this case the client is outside the firewall and the server within. The *direction* column indicates the flow of packets through the firewall:

- < Indicates request arrives on the external firewall interface from the client
- > Indicates, request arrives on the internal firewall interface from the server to the client

Cmd ID	Direction	Source Port	Destination Port
2	<	1206	2119
	>	1048	1204
	>	1049	1204
	>	1050	1205
	>	1051	1205
3	<	1210	2119
	>	1055	1208
	>	1056	1208
	>	1057	1208
	>	1058	1209
	>	1059	1209
4	<	1212	2119
5	<	1214	1061
6	<	1219	2119
	>	1066	1217

	>	1067	1217
	>	1068	1218
	>	1069	1218
7	<	1220	2135
8	<	1225	2119
	>	1073	1223
	>	1074	1223
	>	1075	1224
	>	1076	1221
	>	1077	1224
9	<	1234	1025

Table 5: Example of Network Traffic for each of the Globus Commands

For example, when command 2 is initiated, (`globus-job-run <hostname>/bin/echo Hello World`) the following TCP connection requests occur:

- a. The client (source port 1206) contacts the server (on the destination port 2119: `globus-jobmanager`) via the firewall external interface
- b. The server responds with a series of 4 TCP connection requests back to the client. The server source ports are selected incrementally, while the destination ports at the client are 1204 and 1205 (in this instance).

Note there is only one connection request for command 4 (from client to server), the `url` is returned in a reply packet (Also note the port included in the `url` is determined at the server). The connection in command 5 (required to read the batch job status) connects to the port identified in the `url` returned by command 4. In command 7, the GRIS search is synchronous; the established TCP connection is used to return the requested data. In command 9 the destination port is determined at the server when the user logs in and invokes a `globus-gass-server` to serve his client's request for data transfer.

This part of the investigation revealed that an initial assumption; not allowing TCP *connection* requests on unprivileged ports, was incorrect. Table 5 clearly demonstrates that if connection requests on unprivileged ports are blocked then commands 2,3,6,8,9 will fail when the client is located within the firewall and the server outside. This is significant because unlike the well-known Globus service ports (such as `gatekeeper` and `MDS`) usage of these other ports varies and will be unknown in advance (although a range can be defined).

6 Globus Port Range

In order to utilise a restricted range of open ports through the firewall (in the following example; ports 3000 to 6000), the Globus Toolkit must be constrained to operate within this range. For example when TCP connection requests are initiated in response to the commands 2,3,6,8 in Table 1. This port restriction is achieved through the use of the `GLOBUS_TCP_PORT_RANGE` environment variable. This variable must be set in the *environment* in which the Globus *command* or *daemon* executes.

6.1 Port Security

If external TCP connection requests on all unprivileged ports are permitted to pass through the firewall, then the `GLOBUS_TCP_PORT_RANGE` restriction is not required. However, it is anticipated that many Grid sites will take a cautious attitude by providing a restricted range of ports.

Furthermore, restricting the Grid hosts permitted to receive incoming connections at the firewall would enhance such an approach. Equally, incoming Grid hosts might be filtered by the firewall; but in a large Grid environment this could result in a very large number of packet filtering rules to accommodate all permitted external Grid nodes.

7 Firewall Configuration

The following firewall configurations were used for testing Globus in combination with a range of open unprivileged ports.

7.1 Firewall configuration 1

This is a client only configuration. The node behind the firewall is only regarded as a Globus consumer. Ports in the firewall are blocked preventing incoming requests on the Globus service ports. The original configuration did not allow connection requests on the unprivileged ports. However, after examining the flow of Globus command connection requests, this requirement was relaxed to include a small number of unprivileged ports. The highlighted text in the figure below indicates the changes made to the incoming port address range.

```
RULE SET #1 Daresbury Configuration
-----
# This configuration is intended to provide the functionality observed while at
# Daresbury lab.
# These rules allow hosts inside the firewall to initiate communication with
# the outside world. (basically all outgoing packets are free to pass out) TCP
# connections can be established from inside and the replies allowed back in.
# TCP connection attempts from outside are prevented ( unless they arrive on a
# specified range of unprivileged ports - included as a later addition)
# The ICMP protocol is not subject to restriction.
# UDP for DNS is allowed in.
# IP spoofing on external interface is blocked
#
# All other incoming protocols are blocked.

# INPUT CHAIN, POLICY IS DENY by default (Drop all packets that do not match
# any rule.)
ipchains -P input DENY

#Allow packets on the local loopback adapter
ipchains -A input -i lo -j ACCEPT

# Accept connection requests on a specified range of unprivileged ports
ipchains -A input -I eth1 -s 0/0 -d 0/0 3000:6000 -p TCP -y -j ACCEPT

# Log all incoming connection requests
ipchains -A input -I eth1 -p TCP -s 0/0 -y -l

# Only allow TCP connections in through the external interface if they are not
# connection request packets. ie the TCP packets are in response to a request
# initiated from inside the firewall.
# Applies to all source IP addresses
```

```

ipchains -A input -i eth1 -p TCP -s 0/0 ! -y -j ACCEPT

# Default input chain policy is deny. we need to remove restrictions on
# the internal network interface. Allow all packets from internal network
# unrestricted access
ipchains -A input -i eth0 -s 0/0 -j ACCEPT

# Default policy is DENY.
# ICMP is not restricted for the moment. ALLOW ICMP. Allow on all interfaces.
ipchains -A input -p ICMP -s 0/0 -j ACCEPT

# Allow DNS lookups. We could specify from which sources DNS packets will
# come from. Leave open to any host for the moment. 42 is service num for DNS
ipchains -A input -p UDP -i eth1 -s 0/0 -d 0/0 42 -j ACCEPT

# Prevent IP spoofing on the external network interface (eth1)
# Don't want anyone pretending to be from my internal network.
# Log and DENY these packets
ipchains -A input -i eth1 -s 192.168.100.0/24 -l -j DENY

# If no rule is matched when we get here then the packets (any protocol/ any
# source) get DENIED (Dropped with no ICMP destination unreachable reply!)

```

Figure 2: Firewall configuration 1

7.2 Firewall configuration 2

This configuration is derived from configuration 1. In addition the relevant ports have been opened to receive incoming communication requests for each of the Globus Services (see Table 1). All incoming TCP connection requests are logged by default while all outbound communications remain unrestricted.

```

# This configuration is based on functionality observed while at
# Daresbury lab. In addition Globus ports have been opened in conjunction with the
# base Daresbury configuration.
# These rules allow hosts inside the firewall to initiate communication with
# the outside world. (basically all outgoing packets are free to pass out) TCP
# connections can be established from inside and the replies allowed back in.
# TCP connection attempts from outside are prevented.
# The ICMP protocol is not subject to restriction.
# UDP for DNS is allowed in.
# IP spoofing on external interface is blocked
#
# All other incoming protocols are blocked.

# INPUT CHAIN, POLICY IS DENY by default (Drop all packets that do not match
# any rule.)
ipchains -P input DENY

#Allow packets on the local loopback adapter
ipchains -A input -i lo -j ACCEPT

# Pin open a range of ports for incoming connection requests
ipchains -A input -I eth1 -s 0/0 -d 0/0 3000:6000 -p TCP -y -j ACCEPT
# Log all incoming connection requests
ipchains -A input -I eth1 -p TCP -y -l

# Only allow TCP connections in through the external interface if they are NOT
# connection request packets. ie the TCP packets are in response to a request
# initiated from inside the firewall.
# Applies to all source IP addresses
ipchains -A input -i eth1 -p TCP -s 0/0 ! -y -j ACCEPT

# Default input chain policy is deny. we need to remove restrictions on
# the internal network interface. Allow all packets from internal network
# unrestricted access
ipchains -A input -i eth0 -s 0/0 -j ACCEPT

# Default policy is DENY.
# ICMP is not restricted for the moment. ALLOW ICMP. Allow on all interfaces.
ipchains -A input -p ICMP -s 0/0 -j ACCEPT

```

```

# Allow DNS lookups. We could specify from which sources DNS packets will
# come from. Leave open to any host for the moment. 42 is service num for DNS
ipchains -A input -p UDP -i eth1 -s 0/0 -d 0/0 42 -j ACCEPT

# Prevent IP spoofing on the external network interface (eth1)
# Don't want anyone pretending to be from my internal network.
# Log and DENY these packets
ipchains -A input -i eth1 -s 192.168.100.0/24 -l -j DENY

##### Globus port additions to the base configuration #####
# All outbound traffic is allowed from within the firewall in this configuration, so
outbound
# connections do not need opening. Log all incoming connection requests.

# Allow incoming TCP connection requests on the external interface to the Gatekeeper
(port 2119)
ipchains -A input -s 0/0 -d 0/0 2119 -p TCP -y -i eth1 -l -j ACCEPT

# Allow incoming TCP connection requests to the GRIS (port 2135 tcp) on the external
interface
ipchains -A input -s 0/0 -d 0/0 2135 -p TCP -y -i eth1 -l -j ACCEPT
# Open the GRIS UDP port as well
ipchains -A input -s 0/0 -d 0/0 2135 -p UDP -i eth1 -l -j ACCEPT

# Allow TCP connection requests to the site GIIS on the external interface
# We should specify the IP address of the
# GIIS, but will leave open to any internal host at the moment for testing purposes
(We use port 2136 for GIIS).
ipchains -A input -s 0/0 -d 0/0 2136 -p TCP -y -i eth1 -l -j ACCEPT

# Allow TCP connection requests to GridFTP on the external interface.
# (2811/tcp: control channel) . Check the data channel ports
ipchains -A input -s 0/0 -d 0/0 2811 -p TCP -y -i eth1 -l -j ACCEPT

# Allow TCP conection requests to GridSSH on the external interface. (22/tcp)
# This will allow requests in for regualr SSH also. Currently destination machine is
not
# restricted.
ipchains -A input -s 0/0 -d 0/0 22 -p TCP -y -i eth1 -l -j ACCEPT

# Allow TCP connection requests to MyProxy (7512/tcp)
ipchains -A input -s 0/0 -d 0/0 7512 -p TCP -y -i eth1 -l -j ACCEPT

# If no rule is matched when we get here then the packets (any protocol/ any
# source) get DENIED (Dropped with no ICMP destination unreachable reply!)

```

Figure 3: Firewall Configuration 2

It is worth noting that the Grid Security Infrastructure Secure Shell (`ssh`), which uses the same port as regular `ssh`, is open for bi-directional communications through the firewall. This implies that regular `ssh` will also operate through the firewall in a bi-directional manner; `ssh` connection requests initiated from hosts outside and within the firewall will be established. The firewall administrator will therefore be required to restrict usage based on source and/or destination network address in order to provide a higher level of security.

8 Issues highlighted during Globus testing

Two primary issues were highlighted during the firewall tests. The first was the failure to obtain batch job status information because the URL, returned during batch job submission, contained an inappropriate port number, therefore failing firewall port restrictions to the server. The second issue related to the occasional miss numbering of ports, causing TCP connection requests, in response to client commands, to fail at the firewall.

8.1 Batch job status information

While batch job submission and result retrieval worked correctly, the batch job status command, `globus-job-status`, (command 5 in Table 4) failed when the server was located behind the firewall.

An examination of Table 5 reveals that `globus-job-submit` makes a single connection request to the Globus gatekeeper (on port 2119; open on the firewall). The job URL is returned in a response packet within the same session. Therefore, the command completes without hindrance from the firewall. Equally, `globus-job-get-output` initially performs a connection request to the Globus gatekeeper. Job output is subsequently returned to the client using connection requests that adhere to the `GLOBUS_TCP_PORT_RANGE` set at the client. In this case, the client is not located behind another firewall, so the restricted port range returned is not required.

In comparison `globus-job-status` connects directly to the port, specified in the url returned by `globus-job-submit`. The port is set at the server, but initial attempts to restrict the range of ports used failed. In order for this command to work, the `GLOBUS_TCP_PORT_RANGE` variable must be passed to the job manager's environment. While the environment is read by the job manager code, this approach fails because the Globus gatekeeper is started by `inetd` (following a default Globus deploy). Because `inetd` is not a login process, attempts to set the environment fail (for example using `/etc/profile`). Equally, the Globus daemon startup script has no effect. There does not appear to be a way in a *default* Globus install, to pass the port range environment variable to the job manager. Section 8.3 discusses potential solutions to this problem.

Conversely this problem did not arise when the client was located inside the firewall; the batch job status command succeeded every time. Successful operation occurred because

- a. The outgoing firewall ports were unrestricted so the client had freedom to send out connection requests on any port.
- b. The server was not located behind another firewall that might block connection requests on the given port.

8.2 Port miss numbering

Setting the `GLOBUS_TCP_PORT_RANGE` to include a high numbered end port caused problems. For example, the `globus-gass-server` would consistently assign a port one greater than the high-end port defined in `GLOBUS_TCP_PORT_RANGE`. For example if the range were defined as 40000 to 60000, the port number returned after starting `globus-gass-server` would be 60001.

Equally, `globus-rcp` (command 8) often failed when the client was located within the firewall. In responding, the server would try one of the return connection requests on a port one higher than the specified port end range, e.g. 16001.

Thus in both cases the firewall blocked the incoming connection requests which caused the commands to fail. This problem has not yet been resolved, but a workaround has been found by reducing the port range to include a lower numbered port as the range high point. Subsequent testing used ports between a range of 3000 to 6000 with no ill effects. Initial results appear to indicate the size of the range is not a problem, purely the size of the high-end port number. Further investigation is required to determine the full extent and cause of the problem.

It was also found that the `globus-gass-server` port miss numbering could be avoided by passing the `gass-server` the `-p <port number>` switch. This approach is now preferred over setting the port range environment variable on the server.

8.3 Potential Solutions

The failure of the `globus-job-status` command is due to the `globus-jobmanager` allocating a port that conflicts with the server's firewall port policy. In particular, the job manager's default behaviour is to include the next available unprivileged port address within the URL, returned after job submission. Although the job manager does read environment information, it is not possible to pass the `GLOBUS_TCP_PORT_RANGE` environment variable to the job manager, because

- 1) The gatekeeper is controlled by `inetd`, in a *default* Globus deployment. This implies that it is environment of `inetd` that is read; `inetd` is not a login process so therefore mechanisms like `/etc/profile` have no effect on the environment.
- 2) Globus configuration files are not read for port restriction information. Therefore another mechanism must be found to achieve this.

It is entirely possible to log into the relevant server and retrieve batch job status information locally from that server. This implies a remote communication channel (e.g. `ssh`) is available. But this approach is not considered further because it breaks the philosophy of the toolkit.

A number of potential candidates that do exist include:

1. Leave all unprivileged ports open on the firewall.
2. Leave only low numbered unprivileged ports open on the firewall.
3. Start the Globus gatekeeper as a daemon.
4. Modify the gatekeeper.
5. Redeploy using the `-h host` option.

The first two options are simply workarounds at the firewall level. Option 1 is the simplest, but dilutes the strength of firewall protection. In option 2 a number of low unprivileged ports may be opened for incoming connection requests. Initially this approach would work, but in a busy Grid environment, port assignments would soon exceed the range of low numbered ports open at the firewall; it is clear that a mechanism at the server is still required to constrain port usage.

In option 3, the gatekeeper would be specified as a daemon in the `/etc/globus-gatekeepers.conf` file before local deployment. The `GLOBUS_TCP_PORT_RANGE` environment variable could be set and the daemon started within that environment by the root user. The disadvantage of this approach is the additional overhead incurred and the lack of process monitoring. `inetd` reduces the former (puts the process to sleep when not in use) and provides the latter. Because a redeploy is required to achieve this, it is clear that option 5 is potentially a better approach.

Option 4 requires the most effort, but it is reported that some sites (including Sandia) have modified the Globus gatekeeper code to obtain port range information from the gatekeeper configuration file (`<globus-deploy>/etc/globus-gatekeeper.conf`). This information is then used to set the `GLOBUS_TCP_PORT_RANGE` environment variable *before* the job manager is started, thereby forcing `globus_io` to adhere to the specified range. It is reported that the motivation behind this approach was that it was “*seen as a secure method*”. [Johnson01] It is anticipated however, that the majority of sites will prefer not to incorporate non-standard changes to their Globus deployment.

Finally, it has been reported that option 5, redeploying Globus with the `-host` flag produces a job manager shell script wrapper. Environment information can then be inserted into this wrapper. Unlike option 4, non-standard alterations to code and configuration files are avoided; therefore this approach is deemed to be optimal for most Globus installations.

9 Approach to solving Gatekeeper port pinning

9.1 Redeploy

Calling `globus-local-deploy` with the `-host <hostname>` switch causes a Bourne shell wrapper to be created for the job manager when Globus is (re)deployed. The wrapper, `<globus-deploy>/libexec/globus-jobmanager`, is shown in Figure 4 after a successful (re)deployment. The file must be edited manually to insert the `GLOBUS_TCP_PORT_RANGE` environment variable (see highlighted text).

```
#!/bin/sh
GLOBUS_HOSTNAME=154
export GLOBUS_HOSTNAME
GLOBUS_TCP_PORT_RANGE=12000,16000
export GLOBUS_TCP_PORT_RANGE
exec /opt/globus1.1.3/libexec/_globus-jobmanager "$@"
```

Figure 4: Wrapper to `globus-jobmanager`

During the deployment process the job manager binary is moved to `<globus-deploy>/libexec/_globus-jobmanager` as denoted in Figure 4.

On starting Globus after the deployment, the test machines reported the error,
`/bin/sed: can't read /opt/globus1.1.3/etc/globus-jobmanager.conf: No such file or directory.`

While the file does indeed exist, it is in fact 0 bytes long. Figure 5 depicts the contents of the same file present on another system, which was not deployed using the `-host` flag. This information was used as a template to manually populate the new deployment's 0 byte file.

```
-home /opt/globus1.1.3
-e /opt/globus1.1.3/libexec
-globus-org-dn 'dc=csm, dc=port, dc=ac, dc=uk, o=Grid'
-globus-gatekeeper-host 'marge.csm.port.ac.uk'
-globus-gatekeeper-port '2119'
-globus-gatekeeper-subject '/O=Grid/O=Globus/CN=marge.csm.port.ac.uk'
-globus-host-dn 'hn=marge.csm.port.ac.uk, dc=csm, dc=port, dc=ac,
dc=uk, o=Grid'
-globus-host-cputype i686
-globus-host-manufacturer unknown
-globus-host-osname linux
-globus-host-osversion 2.2.17-33.beosmp
```

Figure 5: Example contents of globus-jobmanager.conf

Further investigation also revealed another 0 byte file, `<globus-deploy>/etc/globus-services`, was causing client job-submission requests to fail with the error:

“GRAM job submission failed because the gatekeeper failed to find the requested service (error code 97)”.

This issue was resolved by editing the file with the necessary job manager information. Figure 6 depicts the information required for the fork job manager.

```
jobmanager stderr_log,local_cred - /opt/globus1.1.3-host-
wrapper/libexec/globus-jobmanager globus-jobmanager -conf
/opt/globus1.1.3-host-wrapper/etc/globus-jobmanager.conf -type fork -
rdn jobmanager -machine-type unknown
```

Figure 6: Fork job manager specification

9.1.1 Redeployment summary

The modifications previously described proved successful in enabling the Globus job manager to reference port range information. Subsequent testing of the Globus batch job commands, in particular `globus-job-status`, completed successfully. Although a full range of ports were not tested, at the 3000-6000 port range, the server's job manager consistently returned port addresses within the specified range.

It appears that the `globus-jobmanager` wrapper can also be created manually, avoiding the requirement to redeploy. Although this has not yet been tested, the process appears to be simply:

- a. Move `<globus-deploy>/libexec/globus-jobmanager` to `<globus-deploy>/libexec/_globus-jobmanager`.
- b. Create the wrapper as in Figure 4. Ensure it is world read and executable.

- c. Insert the `GLOBUS_TCP_PORT_RANGE` environment variable and export.

These steps may prove to be substantially quicker (if no other issues arise) than performing the redeployment.

10 Globus firewall port range summary

The Globus Toolkit has been tested with two firewall port configurations utilizing a method favoured by the Globus project. A range of unprivileged ports were opened at the firewall and an environment variable used to direct Globus network communications to this open range.

The first ‘Grid consumer-only’ configuration was used to investigate the success of Grid client operation from inside a firewall. The second firewall configuration was used to test the operation of a server both inside and outside the firewall.

Two primary issues were highlighted. The first was the failure to obtain batch job status information, due to the assignment of a job URL with an inappropriately numbered port. Although assigned by the server’s job manager, firewall port restrictions were ignored. This issue was resolved by creating a shell wrapper to the `globus-jobmanager` and assigning port restrictions within the wrapper.

The second issue related to the occasional miss numbering of ports, causing TCP connection requests, in response to client commands, to fail at the firewall. Errors were observed across a number of port range settings. Further investigation is required to determine the cause of this problem. However, the range of 3000 to 6000 was consistently used during testing without error.

It was found that `globus-gass-server` port miss numbering could be avoided by specifying the port number on the command line when the server was started, instead of relying on the `GLOBUS_TCP_PORT_RANGE` environment variable.

If both these issues are taken into account, the necessary firewall ports opened, and the `GLOBUS_TCP_PORT_RANGE` variable exported to the client’s environment, then the Globus toolkit will execute correctly through a firewall using this mechanism.

11 The Nexus Proxy

The Nexus Proxy [nexus-proxy01] is designed to extend the Globus Toolkit communications infrastructure to allow Globus interaction through a firewall. In particular the Nexus Proxy relays *“TCP communications to provide a communication mechanism beyond the firewall and between nodes which have local IP addresses”*. [Tanaka01] Specifically this means that the number of ports that must be opened at the firewall are greatly reduced.

11.1 Background

“The basic mechanism of the Nexus Proxy is similar to the SOCKS protocol. Since the SOCKS protocol does not support the handling of passive open sockets however, we can not utilize SOCKS as a proxy server in the Globus system” [Tanaka01]

The Nexus Proxy comprises of an inner server and an outer server. The inner server resides within a site, and the outer server resides externally to the site and firewall. Both proxies are daemon processes. The Nexus Proxy provides modifications to the Globus source code causing proxy enabled versions of `connect()` and `bind()` to be used. Additional environment variables and proxy server address information is required to utilise the Nexus Proxy infrastructure. When a client process from within the firewall tries to connect to a remote host (or bind to a socket), a relay request is transmitted to the Nexus Proxy outer server; the client does not call the regular `connect()` or `bind()` functions directly.

“In the Nexus Proxy system, only the communication port from the outer server to the inner server must be opened in advance, i.e. the firewall must be configured to open the port” [Tanaka01]. Globus communication utilising only this communication channel promises to provide a useful mechanism to firewall administrators wishing to purely open the minimum number of ports possible in order to allow Globus operation.

11.2 Version information

The Nexus Proxy is reported to support Globus 1.1.3 and 1.1.4. The maintainers highlight that while MPICH-G on Globus 1.1.3 works correctly; MPICH-G2 on Globus 1.1.4 currently does not. However, it has also been reported that software maintenance has recently re-started, so issues such as this may soon be resolved.

11.3 Test configuration

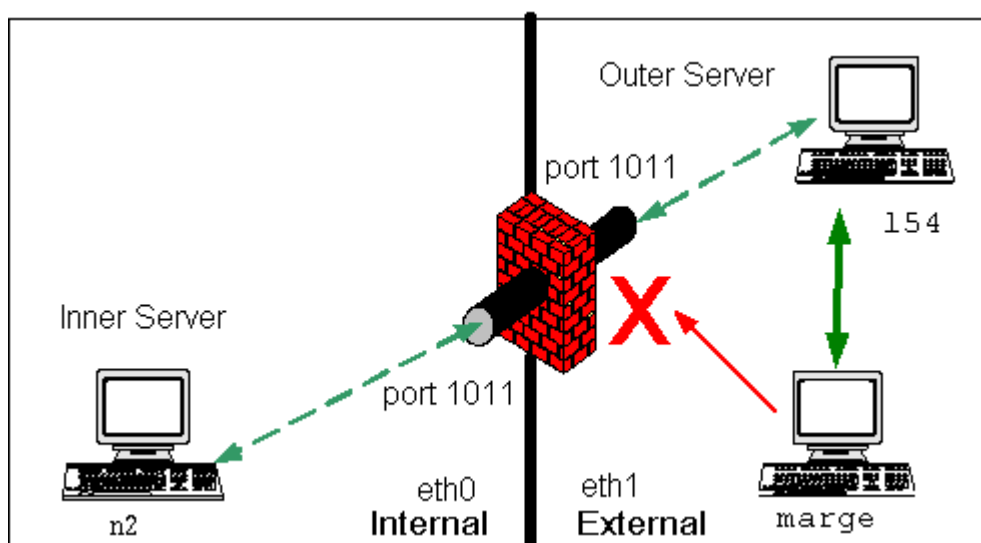


Figure 7: Nexus Proxy test environment

The following conditions and requirements, detailed in the Nexus Proxy installation guide [Tanaka01] were met for the following tests:

- **Condition**

Globus resources are placed within the firewall. The firewall is a deny based firewall; all communication ports are closed for both directions. In fact the firewall is purely logging connection requests so that network interactions can be viewed.
- **Requirements**
 - a. The Nexus Proxy outer server is required to run outside the firewall
 - b. The Nexus Proxy inner server is required to run inside the firewall
 - c. Any port on the outer server must be permitted to connect to the specific Nexus Proxy daemon port on the inner server.

Both systems n2 and 154 are installed with RedHat Linux 6.2 (kernel: 2.2.14-5.0, i586) and deployed with the Nexus Proxy patched version of Globus1.1.3. In the following tests, commands were launched from either 154 or n2 to the opposite machine. The firewall was only logging TCP connection requests on the external firewall interface (both INPUT and OUTPUT chains); no filtering restrictions were active.

Due to the problems encountered, initial testing centered on the use of only two Globus commands:

- a. Interactive submission of the command `/bin/hostname`.
- b. A `grid-info-search` to return the distinguished name and operating system of Globus compute resources known to the local target machine's Grid Information Service (GIS).

11.4 Installation

The Nexus Proxy provides additional libraries to the Globus Toolkit and therefore must be installed on the target machine *before* Globus is compiled. Any existing Globus deployment must be removed, the Nexus Proxy source compiled and the Globus source patched with the Nexus Proxy enhancements. Thereafter the Globus toolkit is built and deployed in the usual way, with the exception that some additional Nexus Proxy options must be added to `globus-install`. The installation is straightforward and trouble free.

The only issue raised throughout the whole process was the absence of a referenced file (`/home/garry/nexus-globus/globus/InformationServices/mds/services/grid-info-site-backend.in`) while applying the Nexus Proxy patch to the Globus source. The default responses were given to the patch utility (`Assume -R [N]` and `Apply anyway? [N]`) causing the patch to be skipped. The remaining patches were applied without further issues.

11.5 Operation

To use the Nexus Proxy, the outer server daemon (and inner server if required) must be invoked. A configuration file for each daemon is used to specify the address range from which the Nexus Proxy is permitted to accept bind and connect requests.

However, Nexus Proxy functionality does not become active unless the environment variables `NEXUS_PROXY_OUTER_SERVER` and `NEXUS_PROXY_INNER_SERVER` are assigned within the user environment from which Globus jobs are executed. These variables reference the server hostname and port for the associated inner or outer server daemon. If the environment variables are not set in the current environment, then the normal Globus communication mechanisms are employed. Wrapper scripts to applications (or C function calls, e.g. `putenv`, embedded within the application) can be used to populate the environment variables with appropriate Nexus Proxy daemon information.

11.6 Testing

11.6.1 Initial comments

During testing, data packets going to non-designated ports were logged over the firewall. Therefore, in addition to the user environment, the Nexus environment variables were exported to the Nexus Proxy daemon environment space on both the inner and outer servers as a precaution.

The Nexus Proxy installation manual denotes that the inner server is optional if the outer server can directly communicate with the computing resources [Tanaka01]. This implies the related environment variable is optional too. However the following error is issued if the inner server variable is not assigned (and the outer server variable is) when a Globus job is submitted:

```
t0:p1790: Fatal error: tcp_init(): globus_io_tcp_create_listener()
failed
```

11.6.2 Test 1: globus-job-run

11.6.2.1 From outer server to inner server

Job submission from 154 (outer server) to n2 (inner server) consistently failed due to mutual authentication failure. See Figure 8.

```
[garry@154 garry]$ globus-job-run n2 /bin/hostname
GRAM Job submission failed because authentication failed:
  GSS status: major:000f0000 minor: 00000000 token: 00000000
  GSS_S_UNAUTHORIZED - wrong gatekeeper or service
  Function:gss_init_sec_context Reason:Mutual authentication
failed
  Expected target subject name="/CN=host/154.csm.port.ac.uk"
  Target returned subject name="/O=Grid/O=Globus/CN=n2.dsg-
cluster.csm.port.ac.uk"
  (error code 7)
```

Figure 8: Nexus Proxy Mutual Authentication Failure

The error output was unexpected. While the target name returned is correct, the expected target subject name is not; the name refers to the submitting host and not the target specified on the command line. Furthermore, the firewall log reveals that the client tried to contact the server (n2) on port 2119. The Nexus Proxy environment variables were exported correctly before `globus-job-run` executed.

Under the normal version of Globus this command executes correctly. However, removing the environment variables to confirm this caused the following fatal error:

```
t0:p1673: Fatal error: tcp_init(): globus_io_tcp_create_listener()
failed
```

In fact all subsequent job execution attempts caused this error to occur, until the client was restarted. Thereafter, the command succeeded as expected under normal Globus operation.

11.6.2.2 From outer server to localhost

The command was issued again, but this time locally on the submitting host (the outer server in this case). As noted above, the 'expected target subject name' was set to the distinguished name of the local host. Executed locally, `globus-job-run` completed successfully returning the hostname '154'.

It interesting to note that even though the command executed locally, traffic still passed over the firewall (Table 6). Connection requests from 154 use the designated port 1011, however, return connections include 1049 and 1053. If the firewall were blocking all but the designated Nexus port (1011) (for *incoming* and *outgoing* traffic) then this interaction would have failed. Clearly, setting the environment variable in the inner server's daemon environment does not result in port restriction on connections established from inside the firewall.

From	To
192.168.1.4:1063	192.168.100.2:1011
192.168.100.2:1028	192.168.1.4:1049
192.168.1.4:1065	192.168.100.2:1011
192.168.100.2:1029	192.168.1.4:1049
192.168.1.4:1067	192.168.100.2:1011
192.168.100.2:1030	192.168.1.4:1053
192.168.1.4:1069	192.168.100.2:1011
192.168.100.2:1031	192.168.1.4:1053

Table 6: Nexus Proxy - globus-job-run localhost - firewall traffic

11.6.2.3 From inner server to outer server

Executing the same command from the inner server (n2) to the outer server (154) caused globus-job-run to receive the following error:

```
[garry@n2 garry]$ globus-job-run 154 /bin/hostname
GRAM Job submission failed because the job manager failed to open
stdout (error code 73)
```

The TCP connection requests logged over the firewall took the form illustrated in Table 7. The initial four requests were directed to port 1011 on the outer server. However then a connection to outer server port 2119 was established, followed by connections to ports 1116, 1114 and 1112. The only connection request from the outer to the inner server arrived on port 1113. Even if the firewall were configured to allow outgoing connection requests on 1011, 2119 and the range of 111n ports, the whole interaction would potentially have failed due to the incoming request on port 1113 at the external firewall interface. Redirecting stdout and stderr to file (relative to the job execution directory) caused the command to hang.

From	To
192.168.100.2:1044	192.168.1.4:1011
192.168.100.2:1046	192.168.1.4:1011
192.168.100.2:1048	192.168.1.4:1011
192.168.100.2:1049	192.168.1.4:1011
192.168.100.2:1050	192.168.1.4:2119
192.168.1.4:1120	192.168.100.2:1113
192.168.100.2:1051	192.168.1.4:1116
192.168.100.2:1052	192.168.1.4:1114
192.168.100.2:1053	192.168.1.4:1112

Table 7: Nexus Proxu - globus-job-run - TCP connection requests from the inner to the outer server

11.6.2.4 From inner server to local host

Finally, globus-job-run from inner server to localhost caused the following error:

```

[garry@n2 garry]$ globus-job-run n2 /bin/hostname
GRAM Job submission failed because authentication failed:
  GSS status: major:000f0000 minor: 00000000 token: 00000000
  GSS_S_UNAUTHORIZED - wrong gatekeeper or service
  Function:gss_init_sec_context Reason:Mutual authentication
failed
  Expected target subject name="/CN=host/154.csm.port.ac.uk"
  Target returned subject name="/O=Grid/O=Globus/CN=n2.dsg-
cluster.csm.port.ac.uk" (error code 7)

```

Again, the expected subject name is returned as the `hostname` for the outer server, while the actual returned target name is the inner server's `hostname`. The associated firewall log is depicted in Table 8.

From	To
192.168.100.2:1075	192.168.1.4:1011
192.168.100.2:1077	192.168.1.4:1011
192.168.100.2:1079	192.168.1.4:1011
192.168.100.2:1080	192.168.1.4:1011
192.168.1.4:1145	192.168.100.2:2119
192.168.100.2:1081	192.168.1.4:1144
192.168.100.2:1082	192.168.1.4:1142
192.168.100.2:1083	192.168.1.4:1140

Table 8: Nexus Proxy - globus-job-run - TCP connection requests from the inner server to localhost

11.6.3 Test 2: grid-info-search

The last command to be tested with the Nexus Proxy was `grid-info-search`. The Grid metadata service does not currently use the Globus security infrastructure, therefore, `grid-info-search` requests to either host complete successfully without the mutual authentication issues noted earlier. It was presumed the LDAP port specified on the command line would be tunneled through the nexus proxy port. However, this proved not to be the case. The firewall log reported port 2135 was still used. If the port number is omitted on the client command line then the error "`ldap_bind: Can't contact LDAP server`", is returned as expected.

11.7 Nexus Proxy Summary

The use of a wrapper to set Nexus Proxy environment information implies that the user can simply stage code to the outer server in order to have it execute on a job manager within the firewall.

However, this requires that the user is aware of the correct values for the inner and outer servers. In a large Grid environment it appears that the user *may* potentially

have to remember the inner and outer server settings for each site they connect to, which implement the Nexus Proxy.

This assumption, however, appears flawed; whilst executing a Globus job on the outer server does cause the job manager to perform network communications over the firewall to the inner server, the returned result is, by this definition, incorrect. The network behaviour implies that a job manager on the inner server executes the request. However, the submitted job, namely to display the hostname of the executing server, returns the outer server's hostname.

If it were not for this returned hostname, or the installation documentation examples, one might assume the execution model would be as follows:

1. Submit a job to the outer server.
2. The outer server then passes the job to the inner-server job manager. The job is then tunneled through the specified nexus proxy port to pass unrestricted through the firewall.
3. The inner server job manager executes the job.
4. Results are returned back through the firewall to the server on non-specified ports.
5. The results are returned from the outer server to the caller.

Another issue is the use of non-designated ports for some TCP connections requests issued during the processing of Globus job submission commands (i.e. `globus-job-run`). Furthermore, it appears strange that commands to return Globus resource information are not piped through the inner/out server communication ports. Instead, a connection directly to the target machine's Grid Resource Information Service (GRIS) is requested on the default port of 2135 across the firewall. This implies that additional firewall ports must be opened, in effect contradicting the Nexus Proxy installation guide.

Although many factors are currently unclear and further investigation is required, the potential benefit of the Nexus Proxy is clear. A dialogue has been established with the software maintainer and a bug report issued. When these initial issues are resolved further comment will be possible.

12 Summary

Although initial experiments with the Nexus Proxy were not successful, it is clear that this software has the potential to provide an extremely useful technique to allow Globus to operate through firewalls. Further investigation is required to understand the issues presented in this report in order to achieve a fully working version of the software.

The firewall port-opening approach favoured by the Globus Project has been shown to work, after some minor adjustments, with a default Globus deployment. However, an outstanding issue of illegal port usage (not adhering to port range restrictions), for commands that create a return connection request, still remains. While testing was successful at the 3000 to 6000 port range, it is clear that such a small fixed range is too restricted to be of much value in a production grid environment.

In terms of installation, the Nexus Proxy was straightforward. However, the requirement to compile the Globus Toolkit against the Nexus Proxy's libraries implies that existing Globus installations must be removed, recompiled and finally redeployed. Some Grid sites may prefer the port opening approach, because the amount of work to achieve this is fairly minimal in comparison to the Nexus Proxy. In fact if the shell wrapper to the Globus job manager is created manually (firewall port opening approach), then the effort and disruption required for an existing site is negligible compared to the Nexus Proxy approach.

Furthermore, the operation of Globus commands through the Nexus Proxy will introduce additional latencies. The effects of such latencies and the performance requirements of the inner and outer servers should be considered in order to determine the requirements for sites with high levels of Grid activity.

In terms of network security, the Nexus Proxy requires a minimum number of firewall ports to be opened. This potentially implies enhanced security (or at least an easier to manage firewall port policy) and therefore it is anticipated the proxy methodology will be preferred at sites with a strict firewall policy.

While testing of the Nexus Proxy failed in this instance, it is clear that both firewall techniques have distinct qualities that will be attractive to Grid installations. Which approach a given site would prefer to adopt depends very much on that site's specific requirements (i.e. security policy, latency issues, disruption of service during transition, etc). However, until the issues experienced with the Nexus Proxy are resolved, this report recommends the use of the Globus "port-opening" technique.

13 References

[Globus01]	The Globus Project, http://www.globus.org , September 2001
[globus-firewall01]	Using Globus/GSI with a firewall, 17/08/01, The Globus Project, http://www.globus.org/security/v1.1/firewalls.html
[globus-qck-gde01]	Globus Quick Start Guide, Globus Software Version 1.1.3 and 1.1.4, The Globus Project, June 2001, http://www.globus.org/toolkit/documentation/QuickStart.pdf
[ipchains00]	Russel, R., Linux IPCHAINS-HOWTO v1.0.8, http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html , July 2000
[Johnson01]	Private communication with Wilbur J Johnson, Sandia Supercomputing Centre, August 2001
[nexus-proxy01]	Tanaka, Y., Download the Nexus Proxy, http://www.apgrid.org/download.htm , May 2001
[Tanaka01]	Tanaka, Y., Nexus Proxy Install Manual, http://www.apgrid.org/src/man.pdf , May 201

14 Appendix A

14.1 IPChains

14.1.1 Operations to work on chains

- 1.Create a new chain (-N).
- 2.Delete an empty chain (-X).

- 3.Change the policy for a built-in chain. (-P).
- 4.List the rules in a chain (-L).
- 5.Flush the rules out of a chain (-F).
- 6.Zero the packet and byte counters on all rules in a chain (-Z).

There are several ways to manipulate rules inside a chain:

- 1.Append a new rule to a chain (-A).
- 2.Insert a new rule at some position in a chain (-I).
- 3.Replace a rule at some position in a chain (-R).
- 4.Delete a rule at some position in a chain (-D).
- 5.Delete the first rule that matches in a chain (-D).

14.1.2 Specific options

Source address: -s

Destination address: -d

Allow icmp packets from 127.0.0.1, append to the input rule :

```
ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

When specifying a protocol (TCP or UDP), a range of port addresses can be given:

i.e.: -p TCP -s 0.0.0.0/0 6000:6010

or a single port: -p TCP -s 0.0.0.0/0 :1023

or use 0/0 to specify any IP address

To allow TCP reply packets through the firewall, block the TCP packets that request a connection (i.e. SYN flag set)

```
-p TCP -s 192.168.1.1 ! -y : rule is only valid for packets that are not a connection request.
```

use -i to specify the port a rule will apply to i.e. -i 192.168.1.100

use -l to log packets firing a rule.

Accounting rules keep count of the number of packets handled by a rule i.e.

```
ipchains -A input -s 192.168.1.1
```

This is purely for information only. No blocking is performed. This rule keeps track of all the packets from 192.168.1.1

Use ipchains -L -v to view the counters for each rule.

Use ipchains -Z <chain name> to zero the counters for a chain.