

Firewall Traversal in the Grid Architecture

Jefferson Tan, David Abramson, Colin Enticott

Faculty of Information Technology

Monash University

Melbourne, Australia

Email: Jefferson.Tan@monash.edu, David.Abramson@monash.edu, Colin.Enticott@monash.edu

ABSTRACT

Computational grids have been at the forefront of e-science supercomputing for several years now. Many issues have been settled over the years, but some may only appear to be so. One of these issues is firewall traversal. Many solutions have been proposed and developed. We have developed two such solutions ourselves: Remus and Romulus, but like all other solutions, they are limited in application. Others are working on proposed standards or solutions based on existing Internet standards and RFCs. However, we still have production-level grids that instead operate their grid resources on an open firewall policy. Some propose moving grids on top of peer-to-peer networks and/or overlay networks, or rebuilding grids on top of clouds instead. Existing grid infrastructures have not rushed to follow either path, however, as the required changes will take considerable effort and cost for currently running systems. This paper investigates the problem and offers a different proposal: a minor revision to the grid architecture. In order to support what we propose, we will look at several proposed solutions and identify their limitations. We also classify them into two distinct approaches, and discuss how each one is not by itself sufficient for all situations. Then we shall show that a slight improvement to the grid protocol architecture provides a multi-pronged *architectural* solution.

Keywords—grids; architecture; networks; protocols; security

I. INTRODUCTION

The Grid is “a distributed computing infrastructure for advanced science and engineering” [1], which has moved up to the forefront of e-science today. Scientific instruments, computers, high-capacity storage devices, data collections and applications from all over the world are being pooled by independent organizations to facilitate collaborative projects. Given the global characteristics of these projects, there is a heavy reliance on networks. Apart from desktops, entire clusters of high-performance machinery would transmit data from proprietary networks, with varying topologies and levels of security, via the Internet. Today’s networks run on standards that make it possible to transcend the heterogeneous nature of computing today. However, some aspects of heterogeneity cannot be so easily resolved, particularly when it comes to diverse network topologies and security policies. One issue continues to cause problems in this regard: *firewall traversal*. This paper focuses on how the architecture must change to resolve such problems.

A. The Grid Architecture

A predominant middleware that is the basis for many grids today is **Globus** [2]. Foster et al. presented its architecture in a seminal paper on the “anatomy” of grids that is organized as a layered protocol stack. Figure 1 illustrates a fundamental aspect of its design: a seamless integration into any existing TCP/IP network. The advantage of such an approach is that it avoids having to use non-standard means of communication. After all, grids build on top of existing computational infrastructure, which are already connected via TCP/IP networks. This has made it almost elementary for grids to grow out of the existing computational and network resources of universities around the world. On the other hand, as many have by now discovered the hard way, the generic nature of such standards can work both ways. While it has made it possible for diverse platforms to join grids, as long as they conform to TCP/IP standards, it has also allowed them to vary in ways that have caused difficulty, particularly in topology and security. *Firewalls* are software-driven devices that protect networks in different ways, primarily by packet filtering [3]. They typically predate participation in grids, and easily cause difficulties because they operate on policies that usually conflict with the resource sharing thrust of grids. Computational clusters are ordinarily oriented towards local use, and making them accessible to external entities is no trivial matter. Figure 1 does not show that default policies to protect these resources dictate how firewalls, operating at the Internet Layer of the TCP/IP stack, restrict access even to specific TCP ports on protected resources. The same can be said about *topological* characteristics of the network. Computational clusters may be connected to a private network that uses the non-routable addresses reserved by the Internet Assigned Numbers Authority (IANA) for that purpose. *Network Address Translation* (NAT) hides a private network behind one point of access, the *NAT gateway*. Access to external networks via NAT is generally restricted to locally initiated connections. This paper does not deal with NAT problems, but it is worth noting the similarity of these issues, both arising from Internet Layer characteristics or policies that were not originally designed to accommodate grid computing.

This is not simply a Globus problem. As long as the grid relies on TCP/IP networking, firewalls will cause difficulties. For example, with Unicore [4], a minimal setup of allowing only one service port cuts performance by half.

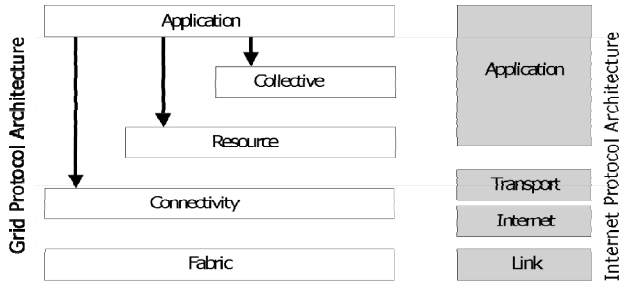


Figure 1. The five-layer protocol grid architecture (Foster et al [1]).

The middleware gLite [5] also requires a generous range of TCP ports. Grid communications should be reasonably open, because each port not only allows one stream of communication, but also a link to one more computations. Limiting the number of accessible ports also restricts concurrent transactions. We do not advocate compromising security, but that is happening now, with administrators forced to relax security for grids. We need communications to be open while maintaining good security. This paper is about an architectural means of achieving this.

B. A Classification of Grid Applications

We characterize grid applications as *networked* or *batched* to illustrate how these classes of applications are affected by NAT and firewalls differently. For *batched grid applications*, jobs are submitted, they execute, then results are returned to the launch node. These applications do not themselves use the network. Job management and file transfer operations are through grid function calls, e.g., a job submission. On the other hand, *networked grid applications* use the network directly. They may still use grid transactions to launch jobs, but must then either allow incoming communications or use the network to communicate with remote components. One example is **Nimrod/G** [6, 7], a middleware that lets users run parameter sweep experiments on resources from different networks and grids. Nimrod/G is slightly different, launching *job agents* on the compute nodes, and the agents then pull jobs (if any) down for execution. These agents communicate with Nimrod/G server components, and vice versa, but firewalls and NAT will prevent this. The Globus toolkit itself is a networked grid application, and it also uses bidirectional communication that can be hampered by firewalls and NAT [8-10]. GridFTP is the primary file staging protocol of Globus, and its operation requires the ability to initiate connections from either side, in either direction. Again, this can be hindered by firewalls and NAT. With the very basis for grid infrastructures encountering such difficulties, the significance of finding a good solution cannot be understated: even basic grid applications are affected if the grid middleware itself is hindered.

Where is the problem? Figure 1 shows the Application Layer of the grid protocol stack at the very top, relying on the Connectivity Layer below in order to use the grid appropriately. To the right, this layer corresponds with the Transport and Internet layers of the TCP/IP architecture, which means that restrictions imposed at the Internet layer

by firewalls will also affect these applications. Therefore, we make this point: issues of firewall traversal should be seen as being *below* grid applications.

C. Firewalls and NAT

In the protocol stack of Figure 1, note that there is no intention for grid protocols to circumvent the TCP/IP stack. In fact, it may be clearer to think of the Transport, Internet and Link layers as sitting beneath the grid's Connectivity Layer. This explains the reason for this paper: certain components of any given TCP/IP network can impede grid connectivity, such as firewalls and NAT. Internet Protocol (IP) firewalls in particular will stand at the borders of IP networks, and properties of a passing packet, i.e., an IP datagram, are inspected and compared against a set of rules. These rules are usually organized in *chains*, and each rule presents conditions that a packet may satisfy, and appropriate action to be taken if so. The firewall subjects the packet to an evaluation process that ends with a decision on whether or not to allow the packet to continue. These rules are based on security policies designed and implemented independently by each organization. Such measures are no longer optional due to growing trends in directed attacks from across the Internet. In NAT, the gateway allows hosts in the private network to reach past the gateway by sharing one or more IP addresses and transport (TCP or UDP) ports accessible via that gateway. The hidden nodes use IP addresses that are reserved for private networks, and are therefore not accessible, because routers will not forward packets destined for such addresses. One might argue that the primary motivation is to minimize the number of leased public IP addresses. Another is to secure the network, since nodes in the private network have no public IP addresses. Packets cannot possibly address them, therefore, without explicit means of exposing them through inbound NAT settings or protocols such as Universal Plug-n-Play (UPnP).

Both firewalls and NAT can disrupt grids because they operate at the Internet and Transport layers, affecting whether or not packets get across. A typical firewall restricts inbound access as fostered by this basic security policy: *reducing exposure*. In TCP/IP terms, this usually means a blanket rule to block all incoming access except for a few authorized and necessary services, and perhaps only from authorized hosts or networks. For example, a university might block access to directory services from the Internet: it can only be accessed from within. Moreover, the server hosting the directory services may only allow access to the TCP port that corresponds to directory services. NAT may also cause issues since the nodes within do not have public IP addresses. Some grid applications require initiating connections from either side, while nodes behind NAT cannot accept incoming connections.

Neither firewalls nor NAT should be considered detrimental. They play key roles in the network. There must be a way to enable connectivity in spite of them. The solution is not in relaxing the firewalls, as done in many production grids we are aware of. Globus recommends *approximately 10 ports for incoming connections at the client site for each concurrent user, 20 ports for incoming*

connections at the server site for each concurrent user. This is a considerable relaxation of security, not to be done lightly, but in our experience, many administrators find little choice. We instead propose an architectural solution, a **Virtual Connectivity Layer** [11] between the grid above and the existing network infrastructure below.

Section 2 presents a brief survey of the literature, where we focus on two particularly serious endeavors in this problem domain: the *middlebox communications*, and *firewall virtualization* for grids. We will also mention alternative approaches that leave firewalls alone, such as by tunneling or proxies. Section 3 presents our proposed six-layer architecture, where the extra layer mentioned earlier will allow existing grids to work in harmony with networks that use firewalls and/or NAT. In Section 4 we discuss an implementation of the proposed layer. Here we explore how we have and have not met the intended solution based on the proposed layer. We then present conclusions in Section 5.

II. RELATED WORKS

Many solutions have been developed in the past years dealing with firewall or NAT traversal. We describe several here with the assumption that the underlying network design is left mostly untouched. These solutions loosely fall under two classifications: firewall virtualization and tunneling.

A. Firewall Virtualization

Firewall virtualization has been used to mean different things. It can refer to exposing the firewall as a grid component that can be controlled by grid middleware [3]. The Open Grid Forum (OGF) has been leading the charge in this regard, and is currently working on the Firewall Traversal Protocol (**FiTP**) [12]. One challenge is integration with the Grid Security Infrastructure (GSI). However, the primary concern is at the lower levels: what protocol and semantics will govern how the firewall is controlled by grid software? An IETF (Internet Engineering Task Force) working group has an answer: *middlebox communications* (**midcom**) [13-15]. The working group has had to study how best to design midcom standards, not only in facilitating necessary functions, but also in preserving security.

There is an alternative terminology that associates firewall virtualization with *security contexts* [16], i.e., virtual firewalls. In larger internetworks, it becomes advantageous to virtualize firewalls so that a smaller number of them provide a larger number of choke points, each being a security context. In this paper, we will refer to this concept as *security contextualization*, in order to distinguish it from the grid-specific notion of firewall virtualization, as defined by the OGF. Security contextualization facilitates separate policies for grid-connected networks. Resource sharing is suitable to these networks, but not for the rest of the organization. Security contexts can reduce the conflict between grid objectives and the organization's overall security policy.

Others have developed similar approaches. The Cooperative On-Demand Opening (**CODO**) of firewalls is one [17], and **Dyna-Fire** is another [18]. The objective is the same: there is a base level of security and grid

transactions will be the object of dynamic firewall exceptions that are accommodated as required. The Globus Alliance supports **Fireblade** [19] as an interim solution while OGF's FiTP is being finalized.

The primary difficulty with firewall virtualization is its impact on overall security. Firewall rules are traditionally seen to be almost static, and changes to them should be infrequent. Furthermore, changes are done by hand, which makes each change explicit and, theoretically, more thoroughly thought out. Firewall virtualization as envisioned opens up a can of worms because the firewall becomes dynamic, and changes may be frequent, depending on the level of grid activity. While audit trails will help, the implications on overall security are there. On the other hand, security contextualization has a different set of implications. One of these affects the design of the organization's internetwork, since one of the contexts, i.e., a virtual firewall, will be designated for one or more grid-connected resources. This will at least affect the routing topology, and possibly the IP addressing of grid resources. However, this can be an advantage as it decouples the relatively lenient security policies for grid resources from the more restrictive parts of the organization. It is, however, important to note that the impact of such a move is non-trivial.

Standards have been proposed through the IETF taskforce on midcom [13-15, 20-22]. Of particular interest are semantics that developed in this domain [20]. They are key to standardizing middlebox access. The basic model adheres to the proven simplicity of standard network management, here involving a *middlebox agent* that must satisfy security constraints before the middlebox trusts it. Then this agent is allowed to influence the middlebox, e.g., add or remove a firewall's rules. In RFC 4097 [22], several candidate protocols were evaluated to find an appropriate basis for the midcom protocol. The most suitable appears to be the Simple Network Management Protocol (SNMP) [23], which already adheres to the very simple model of two components: the *management agent* and a *managed object*. It also complies with most of the midcom requirements when dealing with NAT, while more work is required to influence firewalls. In previous work [24], we were able to implement a simple mechanism to use SNMP to change the *access control lists* (ACLs) of a Cisco router. This is presented later in Section 4.2.

B. Tunneling

One alternative to firewall virtualization and/or security contextualization is to leave the firewall alone and facilitate tunnels and proxies instead [25-28]. A *tunnel* is simply a means of encapsulating one protocol within another. A *proxy* is a software component that encapsulates packets and conducts them across, perhaps through a chain of other proxies. The encapsulating protocol provides the tunnel, and should be managed with sufficient control mechanisms for authentication, authorization and monitoring. The encapsulated protocol would be any authorized protocol, in this case, one that conducts grid transactions. The **Nexus Proxy** is one of the earliest solutions developed specifically for Globus [29], but there are others [9, 30, 31]. A general-

purpose tunneling protocol is the secure shell (SSH) protocol [32], which can provide static one-to-one port forwarding tunnels, or dynamic SOCKS proxy connections [25, 26]. In some implementations, SOCKS has the ability to provide tunneling without explicitly reprogramming the encapsulated protocol to use SOCKS. This is beneficial for two reasons: tunneling leaves the firewall alone, but additionally, some implementations of SOCKS will also allow us to leave the communicating applications alone by automatically redirecting connections. In addition to the fact that SSH tunneling is authenticated, managed and can be monitored, this approach has low infrastructural impact. A disadvantage of this approach, however, is that the extra layer of encapsulation increases protocol overhead and slows down data transfer throughput.

C. Other Solutions

As mentioned, the problem can come from the network topology, with restrictions implemented by components of the Internet Layer in the TCP/IP stack, e.g., firewalls and NAT. It can be argued that this is not a strictly grid-specific problem, so non-grid solutions can apply. Peer-to-peer (P2P) networking has been effective in some general-purpose and grid applications. IP over P2P (IPoP) [33] is one solution that uses P2P overlay networks, which are usually capable of firewall traversal in many situations. IPoP intercepts IP traffic and uses a *tap* interface on one host to route the traffic to another tap interface on the remote host. Assuming that the actual IP datagrams proceeding across taps are allowed by firewalls, traversal becomes possible. WOW [34] is another solution by the same group that uses a further virtualization layer on top of P2P overlay networks. These are good solutions provided that the overlay network is permitted across the participating networks. It must also be assumed that tap interfaces and tunneling IP packets through some transport are allowed. The **Smartsockets** [35] library provides for connectivity through communication proxies called *hubs*. In situations where firewalls or NAT do not permit direct connections, smartsockets allow several means of traversal, such as invoking reverse TCP initialization, TCP splicing or routing. Hubs form a *side channel* that can be used by hosts that have access to hubs, either as a control channel to invoke reverse or splicing methods, or to actually perform routing through the channel. This is also good while networks can host hubs to form a side channel, and if applications can use smartsocket libraries. Note that using P2P overlay networks and routed Smartsockets closely resemble a tunneling approach. Discussions in subsequent sections pertaining to tunneling and proxies are therefore pertinent to P2P overlay networks and solutions like Smartsockets.

D. The Best Solution?

The obvious question now is this: which solution works best? Firewall virtualization seems to put the firewall at risk, exposing it as a controllable resource. Tunneling can be messy, requiring the hosting of proxies in between sites, and reduces data transfer performance, as would P2P overlay networks or smartsocket side channels. P2P is not permitted

in many networks, and side channels require the hosting of hubs, which may not be possible nor allowed in many situations. We choose the middle ground by proposing that *no solution fits every situation*, so we propose a flexible approach that can exploit all possible methods selectively.

III. VIRTUALIZED GRID CONNECTIVITY

Figure 2 illustrates the proposed virtual connectivity layer that provides the following advantages:

- we can choose the best traversal mechanism at runtime
- the firewall can be left unchanged
- the applications can be left unchanged
- traversal can be managed, monitored and audited

A middleware that implements virtualized connectivity must implement two components:

1. A *virtual connectivity wrapper* should shield all grid components from the implementation of connectivity. This means that grid middleware such as Globus, and applications such as Nimrod/G, should not have to be reprogrammed to suit a particular transport mechanism.
2. A *library of transport mechanisms* must exist so that the connectivity wrapper above can invoke the most appropriate one at runtime. At present, we have two mechanisms available, and the idea is to populate this library with as many mechanisms as are possible.

These two correspond directly with these principles: to provide an abstraction layer and to provide flexibility. Based on the first, firewalls, NAT gateways and applications should not require substantial reprogramming. The second principle is about being able to use the right mechanism for the right situation.

All existing solutions come at a price, but this can be improved upon through policy design. Administrators of individual domains must decide how they wish to proceed. As they choose different policies, the resulting infrastructure (changes) will allow one solution but perhaps not another. The implications in firewall virtualization are twofold: there should be enough manageability to allow reconfiguration on the demand, and control software must integrate with the

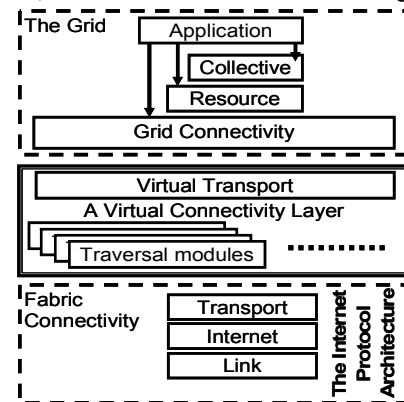


Figure 2. Proposed new architecture with a middle VC layer.

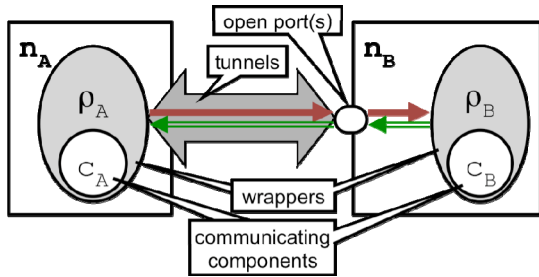


Figure 3. Remus firewall traversal by wrappers, rerouters, and tunnels.

grid security infrastructure. Tunneling assumes that connecting into (or out of) a network is possible through some protocol that supports tunneling. It is expected that every network joining the grid will be unique in certain ways, and it is unlikely that their administrators would be quick to change their topology and policies. A multi-pronged solution makes sense. Considering our point above about making several mechanisms available, the following should be considered representative of solutions of the same class that are found in the literature, with some variations.

Figure 3 illustrates **Remus**, a *RErouting and MUltiplexing System* that uses tunneling [36]. Temporary tunnels convey application connections in either direction. Software components that need to communicate are invoked by *wrappers*, which enable the automatic rerouting. The wrapper decides when to use particular *rerouters*, and how to encapsulate application packets using protocol data that the rerouter can understand. The diagram shows one rerouter within each wrapper, but several rerouters may actually be used. A *rerouting table* is defined beforehand, by which a rerouter is chosen at runtime. For most intents and purposes, every rerouter is a TCP or UDP *proxy*, using particular protocols such as SOCKS or HTTP [25, 37].

A proxy encapsulates packets using a specific protocol and sends it to the destination via a tunnel it is attached to. We usually distinguish between proxy clients and servers, but in this paper, we simply refer to both ends of a tunnel as rerouters, as each rerouter should allow for incoming and outgoing communication. Remus possesses certain aspects:

- bidirectional communication, initiated from either direction, should be facilitated
- multiple tunnels will improve throughput [28, 38]
- the port bound to a rerouter must be accessible
- traffic may be encrypted and compressed
- a tunnel may only be constructed for authorized users
- tunnel activity can be audited
- in case of failure, the tunnel can be reconstructed
- the rerouting table may be updated at runtime

Up to this point, we have only discussed Remus and firewalls, because dealing with NAT is more difficult, particularly when a new connection must be initiated *with* a node behind a NAT gateway. An assumption when dealing only with firewalls is that resources possess global (IP) addresses. Nodes behind NAT have private addresses, so it is not straightforward to address a connection request to such nodes. It is possible to make private IP addresses

accessible via the application-level rerouting tables, but that is not practical. Private addresses typically use the 192.168.0.0-192.168.255.255 range. Networks behind different NAT gateways, perhaps from different organizations, may use the same addresses. We are still working on a tunneling solution to NAT problems.

Figure 4 illustrates a firewall virtualization transport mechanism we have implemented, **Romulus** [11]. It features a firewall agent, a *midcom* agent in the terminology of IETF midcom [13], that receives requests from users or their applications by proxy, and authenticates the requesting entity. If authentication is successful, the agent tells the middlebox to temporarily open a *pinhole* specifically for the requesting entity [15]. The pinhole duration is based on the expiry of the requesting entity's credentials. In many ways, Romulus is of simpler design than Remus simply because there is no rerouting. Connections are direct, although Romulus will not work unless the agent can control the middlebox. It is intended that Romulus will facilitate traversal of NAT gateways as well as firewalls. The solution is in the way the agent will control the middlebox when NAT is in use. This is future work, but it should be achievable since NAT can proceed in both ways, and an arbitrary port on the NAT gateway may be mapped to the intended port on the resource behind it.

Such solutions are known for Globus. An opening across NAT may also be considered a pinhole, and may be opened similarly. There is a limit on the number of ports available on the NAT gateway, even if a pool of several IP addresses is available for NAT, but the solution is possible.

IV. IMPLEMENTATION

This section describes the implementation of the multi-pronged approach described above, using Remus and Romulus, and a virtual connectivity wrapper in Remus.

A. Remus

Figure 5 illustrates how Remus works. The wrapper will use a rerouting table as a basis for rerouting connections via SOCKS proxy. Several may be in use for multiple tunnels. Some may be tunneled as SOCKS/SSH/TCP (TCP containing SSH packets, which contain SOCKS packets, which contain the grid protocol payload). Others may proceed without SSH, purely as SOCKS/TCP. It entirely depends on which SOCKS proxies are referred to. A typical scenario uses SSH as the tunneling protocol, since it is trusted and secure, and OpenSSH supports SOCKS proxy functions. This approach is appropriate when the user is

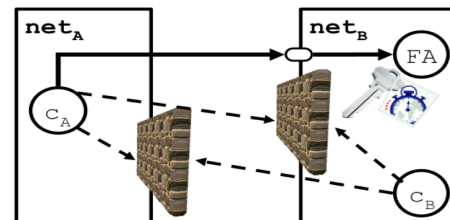


Figure 4. Romulus enables communications normally hindered by firewalls through a prior, perhaps transparent request to a Firewall Agent (FA), which opens pinholes for the requesting component (C_A).

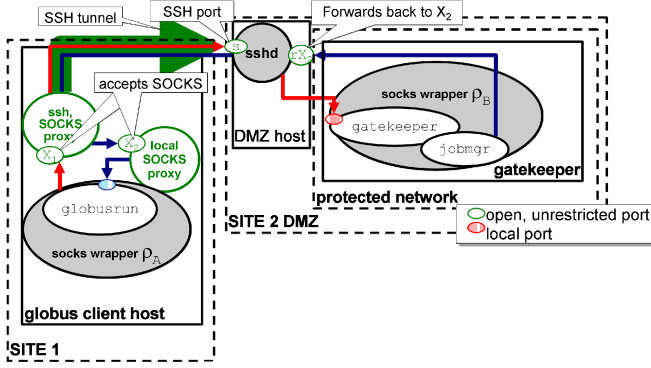


Figure 5. Remus uses SSH tunnels and SOCKS to encapsulate connections transparently. Processes are invoked with wrappers, and child processes inherit the environment automatically.

authorized to connect to the remote site via SSH with a valid user account. The account need not be on grid resources themselves. Many organizations provide a “staging” host outside the firewall that allows SSH access. This is shown in Figure 5 on the DMZ (demilitarized zone) host, which sits outside the protected network. SSH on the staging host will convey connections through to the rest of the organization’s network. Table 1 shows a rerouting table where two hosts run rerouters. The first two entries map a connection by rerouting to Rerouter host 1 on ports 1080 and 1081, depending on the destination of the packet being Host A or B, respectively. The third entry maps to Rerouter host 2 on port 1080 to connect to Host C. These rules are necessary if direct connections to A, B and C are impossible. Finally, other hosts may be accessed directly.

In tests conducted over controlled environments and the Internet, throughput degradations between 6% and 39% were observed [11, 27]. The best performance used a 5:1 ratio, i.e., five TCP connections in one tunnel. The worst was at a 15:1 ratio.

Results were obtained by running simultaneous tests with the Linux tool **iperf**, for as much as 12 hours. Such degradation makes the most impact when data transfer throughput is a factor. However, there are situations where the firewall does not otherwise allow staging in, so even a slow tunneling transport is preferable.

B. Romulus

In Romulus, users explicitly request access on the basis of X.509 certificates in one of two ways: through a secure website, or via GridFTP [24, 39]. Figure 6 illustrates the two client-agent interactions to (1) authenticate the client, then

TABLE I. A SIMPLIFIED REROUTING TABLE.

Source		Destination		Rerouter	
Address	Port	Address	Port	Address	Port
Any	Any	Host A	Globus port range	Rerouter host 1	1080
Any	Any	Host B	Globus port range	Rerouter host 1	1081
Any	Any	Host C	Globus port range	Rerouter host 2	1080
Any	Any	Other hosts	Any	direct	

(2) send pinhole details. If authentication succeeds, the firewall agent inserts a firewall rule (3) to open a pinhole as requested. Grid connections may now proceed directly without impediment through the pinhole (4). The pinhole automatically pertains only to Globus-specific ports. The pinhole is eventually removed when it expires (5), at the expiry of the proxy certificate, or when the user’s certificate is revoked, whichever comes first. Somewhere between steps (3) and (4), the firewall agent updates a pinhole table to record the pinhole identifier and expiry.

The secure website approach is easier, authenticating via username/password. A preferred approach is via X.509 certificate authentication, with the certificate loaded onto the web browser and verified to authorize requests. There are other possibilities as well, such as incorporating Shibboleth single-signon [40]. At present, the user can explicitly post both an IP address or address range, and how long to keep a pinhole open. To comply with the security infrastructure, the duration requested is checked against the expiry time of the user’s certificate.

The GridFTP approach is also straightforward, where the user uploads a proxy certificate, and a successful upload implies authentication. The proxy certificate expiry determines the duration of the pinhole. The IP address of the requester, recorded in GridFTP logs, is used as the basis for the pinhole. The proxy certificate expiry cannot exceed user’s certificate expiry. Romulus has been implemented on two platforms: Unix-based firewalls and Cisco routers with ACLs. For a Unix-based router, standard firewall platforms such as **iptables** or **ipfw** are flexible enough to manipulate. For example,

```
iptables -I GLOBUS -p tcp -m tcp -m comment --comment $f \
-s $IP --dport $port -j ACCEPT
```

This adds a rule to the *GLOBUS* chain of rules to open a pinhole for connections from the IP address *\$IP* to any destination on TCP port *\$port*. The comment *\$f* is a label that identifies each pinhole uniquely. The agent maintains a table to keep track of the intended expiration time of each pinhole, indexed by the pinhole identifier *\$f*. The agent periodically checks the table for expired pinholes and removes them from the firewall when this happens. We experimented with Cisco 2600 routers that implement ACLs. The immediate difficulty was the router’s inability to

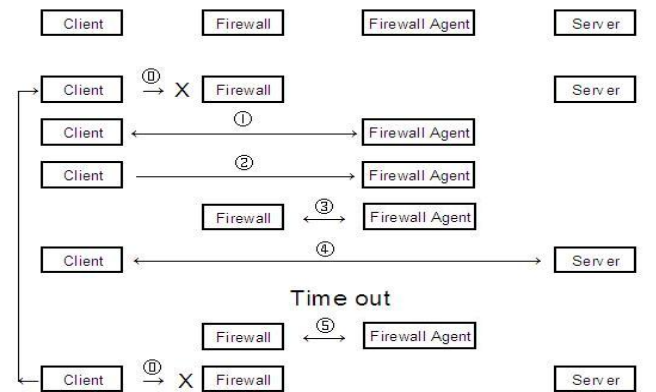


Figure 6. Interactions among the client, firewall agent and firewall [24].

host a midcom agent that incorporates midcom semantics, although SNMP can still be used to prompt the router to use TFTP to pull down new configurations to open or close pinholes. More recently, we are attempting to use better techniques with a Cisco ASA 5510 firewall. However, despite SNMP functionality, its midcom capabilities, e.g., fine-grained ACL control, continue to rely on a web-based interface. We are now trying to integrate certificate-based authentication with RADIUS, which the firewall supports. The web-based interface at present is subject to abuse, although using SSL and X.509 certificates would be a reasonable improvement. The GridFTP interface is not without problems: submitting the proxy certificate is risky.

Romulus' pull model involves some latency. Instead of being directly controlled by the pinhole request, the firewall agent only checks for requests by the minute, using **cron**, the Unix scheduled execution service. Another source of latency is the actual pinhole operation, from instantaneous to a minute, depending on the size of the pinhole table. A persistent table on disk would be more robust, but file I/O latency can be a problem. When invoked, the agent will also check the table for expired pinholes that need to be disabled. Since we did not test for the effect of large pinhole tables on latency, extensive tests should be conducted in the future. The response time from the kernel to actually create or remove pinhole rules was minimal, and it is apparent that the pinhole table would have the most effect on overhead. On the other hand, once the pinhole is open, the actual grid communications thus enabled will not suffer any overhead.

C. The Virtual Connectivity Wrapper

As earlier mentioned, the virtual connectivity wrapper shields the virtual connectivity layer from the grid above it. Neither grid middleware (such as Globus) nor applications (such as Nimrod/G) should make adjustments. This is unlike the approach of solutions like WOW, which require virtual machines. Our approach is to allow clusters and grid middleware to remain unchanged, while we attempt to virtualize only the connectivity across sites. This is successfully implemented in Remus. **Dante** provides SOCKS wrappers via command line utility, **socksify**, or by setting the `LD_PRELOAD` environment variable [41]. Network calls by the wrapped process are rerouted automatically, subject to the SOCKS configuration file, which we use as the rerouting table. The wrapper can thus make decisions to either allow the function call to go through to the operating system unimpeded, i.e., direct connection, or to encapsulate it via SOCKSy. The rerouting table, e.g., Table 1, allows the definition of multiple SOCKS proxies, assumed to already be operational. A similar ability to invoke Romulus has yet to be completed. It should be possible for the wrapper to decide to either automatically send a request to the midcom agent on behalf of the application or user, or the midcom agent monitors incoming traffic at the firewall point and reacts, based on initial certificate-based authentication data sent by the client.

Virtualization, applied only to connectivity, leaves grids and computational resources untouched as much as possible. Platform changes can be expensive. It can be costly to link

new libraries with existing applications. It is expensive to adopt or move services into new (virtualization) platforms, setting up overlay networks, or other changes that carry considerable costs. Something else to consider is that adhering to standards is reasonable. It makes sense to use Internet standards (from the IETF) that can be integrated into future versions of middleboxes.

V. CONCLUSIONS

Given the need to secure our networks, the common approach of relaxed firewalls in grids is problematic. On the other hand, firewall restrictions do hinder grid applications. We reviewed a number of firewall traversal solutions, each limited in application. What we propose is to deal with the issues in the architecture: a *virtual connectivity layer* between the grid and the fabric. With the new layer dealing with firewall and similar issues, we avoid having to reprogram grids and grid applications for that purpose. We illustrated this through the wrapper that works reasonably well in Remus, automatically rerouting grid connections through tunnels and proxies. We also push for a library of several possible transport mechanisms that will hopefully include the most appropriate one for different scenarios. We presented two such mechanisms, Remus and Romulus, which correspond to two classes of firewall traversal: tunneling and firewall virtualization. It should be possible to include other solutions that others have already implemented. However, authentication and authorization should be based on grid security mechanisms.

Future work should deal with NAT. We should also explore extending our work to Unicore and gLite grids, which have similar firewall requirements to Globus. We should perfect the connectivity wrapper so that it can invoke any of the available transport mechanisms as needed. The other crucial task is to extend the ability of Romulus to deal with enterprise-level firewalls with fine-grained ACL updates. This must be done with the proposed IETF midcom standards in mind, based on SNMPv3. Something else that must be dealt with in firewall virtualization is to the pinhole duration. User credentials and proxy certificates can extend for months. While the associated pinhole is limited to the user's particular IP addresses, a more desirable level of security should still limit the duration.

ACKNOWLEDGMENTS

Y. Chang, K. W. Kim and M. Albathan, and J. Alshammari, Q. Zhang, A. Gargote, and E. Erick.

REFERENCES

- [1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15, 2001.
- [2] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," *Journal of Computer Science & Technology*, vol. 21, pp. 513-520, 2006.
- [3] R. Niederberger, W. Allcock, L. Gommans, E. Grünter, T. Metsch, I. Monga, G. L. Volpato, and C. Grimm, "Firewall Issues Overview," *Open Grid Forum Aug 16*, 2006.

- [4] "Unicore website," Distributed Systems and Grid Computing Juelich Supercomputing Centre Institute for Advanced Simulation Research Centre. (<http://www.unicore.eu/>), 2010.
- [5] "gLite website (<http://glite.web.cern.ch/glite/>)," Enabling Grids for E-science (EGEE), 2010.
- [6] D. Abramson, R. Sasic, J. Giddy, and B. Hall, "Nimrod: A tool for performing parametrised simulations using distributed workstations," in 4th IEEE Symposium on High Performance Distributed Computing Virginia, August 1995.
- [7] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: an architecture of a resource management and scheduling system in a global computational Grid," in HPC Asia 2000 Beijing, China, May 14-17, 2000.
- [8] V. Welch and O. Mulmo, "Using the Globus Toolkit with Firewalls," <http://www.clustermonkey.net/content/view/122/32/>, 2006.
- [9] S. Graupner and C. Reimann, "Globus grid and firewalls: issues and solutions in a utility data center environment," Hewlett-Packard Company Oct 2, 2002.
- [10] V. Welch, "FirewallHowTo," Globus Alliance. <http://dev.globus.org/wiki/FirewallHowTo>, 2009.
- [11] J. Tan, D. Abramson, and C. Enticott, "A virtual connectivity layer for grids," in 5th IEEE International Conference on e-Science (e-Science 2009) Oxford, UK, Dec 9-11, 2009.
- [12] R. Niederberger, "FiTP - A protocol draft for dynamic opening of Firewalls," in 25th Open Grid Forum (OGF25) Catania, Italy: Open Grid Forum, March 2-6, 2009.
- [13] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and Issues," IETF, RFC 3234, Feb 2002.
- [14] J. Quittek, M. Stiernerling, and P. Srisuresh, "Definitions of Managed Objects for Middlebox Communication," IETF, RFC 5190, 2006.
- [15] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox communication architecture and framework," IETF, RFC 3303, 2002.
- [16] O. Santos and J. Frahm, "Cisco ASA Security Contexts," in Cisco ASA: All-in-One Firewall, IPS, and VPN Adaptive Security Appliance: Cisco Press, 2005.
- [17] S. Son, B. Allcock, and M. Livny, "CODO: Firewall Traversal by Cooperative On-Demand Opening," in 14th IEEE Symposium on High Performance Distributed Computing (HPDC'05), July 24-27, 2005, pp. 233- 242.
- [18] D.-G. I. (DGI), "Dynamic Firewalls Configuration, Ver. 1.0," D-Grid Integrationsprojekt (DGI) Feb. 2006.
- [19] "Incubator/Fireblade (Wiki page)," Globus Alliance. <http://dev.globus.org/wiki/Incubator/Fireblade>, 2009.
- [20] M. Stiernerling, J. Quittek, and T. Taylor, "Middlebox Communications (MIDCOM) Protocol Semantics," IETF, RFC 3989, 2005.
- [21] R. P. Swale, P. A. Mart, P. Sijben, S. Brim, and M. Shore, "Middlebox Communications (midcom) Protocol Requirements," IETF, RFC 3304, 2002.
- [22] M. Barnes, "Middlebox Communications (MIDCOM) Protocol Evaluation," IETF, RFC 4097, June 2005.
- [23] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)," IETF, RFC 1157, May 1990.
- [24] Q. Zhang, "Firewall virtualization over grids," Master of Network Computing: Monash University, 2009.
- [25] Y.-D. Lee, "SOCKS: A protocol for TCP proxy across firewalls," NEC 2005.
- [26] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS protocol version 5," IETF, RFC 1928, 1996.
- [27] J. Tan, D. Abramson, C. Enticott, "Optimizing Tunneled Grid Connectivity across Firewalls," in 7th Australasian Symposium on Grid Computing and e-Research Wellington, New Zealand: Australian Computer Society, 2009.
- [28] M. Albathan, "Performance study of multi-tunneling grid transactions," Master of Network Computing: Monash University, 2009.
- [29] Y. Tanaka, M. Hirano, H. Nakada, and S. Sekiguchi, "Performance evaluation of a firewall-compliant Globus-based wide-area cluster system," in IEEE International Symposium on High-Performance Distributed Computing 2000 (HPDC 2000), Aug 1-4, 2000, pp. 121-128.
- [30] S. Son and M. Livny, "Recovering internet symmetry in distributed computing," in 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), pp. 542- 549.
- [31] S. Son, M. Farrellee, and M. Livny, "A Generic Proxy Mechanism for Secure Middlebox Traversal," in IEEE International Conference on Cluster Computing (Cluster 2005) Boston, MA, Sep. 27-30, 2005, pp. 1-10.
- [32] T. Ylonen, "SSH -- Secure login connections over the Internet," in 6th USENIX Security Symposium San Jose, CA, July 1996, pp. 37-42.
- [33] A. Ganguly, A. Agrawal, P. O. Boykin, and R. Figueiredo, "IP over P2P: Enabling Self-configuring Virtual IP Networks for Grid Computing," in 20th International Parallel and Distributed Processing Symposium (IPDPS-2006): IEEE, 2006.
- [34] A. Ganguly, A. Agrawal, P. O. Boykin, and R. Figueiredo, "WOW: Self-organizing Wide Area Overlay Networks of Virtual Workstations," in 15th IEEE International Symposium on High-Performance Distributed Computing (HPDC-15) Paris, France: IEEE, 2006.
- [35] J. Maassen and H. E. Bal, "SmartSockets: solving the connectivity problems in grid computing," in 16th International Symposium on High Performance Distributed Computing (HPDC 2007), J. D. C. Kesselman, D. W. Walker, Ed. Monterey, California USA: ACM, 2007, pp. 1-10.
- [36] J. Tan, D. Abramson, and C. Enticott, "REMUS: A Rerouting and Multiplexing System for Grid Connectivity Across Firewalls," Journal of Grid Computing, vol. 7, pp. 25-50, March 2009.
- [37] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," IETF, RFC 2616, June 1999.
- [38] J. Tan, D. Abramson, and C. Enticott, "Optimizing Tunneled Grid Connectivity across Firewalls," in 7th Australasian Symposium on Grid Computing and e-Research Wellington, NZ, Jan. 21, 2009.
- [39] J. Alshammari, "Prototyping firewall virtualization for grids," Master of Network Computing: Monash University, 2009.
- [40] "The Shibboleth website," Internet2. Available: <http://shibboleth.internet2.edu/>.
- [41] "Dante- A free SOCKS implementation," Inferno Nettverk A/S. Available: <http://www.inet.no/dante/>, 2005.