Rerouting Grid Applications Across Firewalls

J. Tan, D. Abramson and C. Enticott

Caulfield School of Information Technology, Monash University

900 Dandenong Rd, Caulfield, Australia,3145
{ jefferson.tan, david.abramson, colin.enticott}@infotech.monash.edu.au

Extended Abstract for Poster

The Grid refers to "a distributed computing infrastructure for advanced science and engineering" [4]. Grid applications require advanced computations through distributed processing, with an explicit focus on supporting *virtual organizations* (VOs). By participating in a VO, participating institutions are able to share and coordinate computations, commonly with the Internet as the underlying communication infrastructure. Unfortunately, on the Internet, there is always the risk of unauthorized access or even sabotage to data, transactions or computational resources. Thus, networks on the Internet are usually protected with firewalls and other measures. This presents a dilemma for VO setups since this reduces the level of collaboration that is possible. Furthermore, current Grid applications often assume that network connectivity is mostly unrestricted. Accordingly, a number of virtual organizations have failed because the member sites have security policies that clash, encountering more restrictions than anticipated in the design of their applications. Moreover, since the participants in a VO are autonomous to one another, their network security measures are likewise quite varied. These issues may be taken together as a *security interoperability* problem [3]. Somewhere in the interconnected networks involved, a mismatch occurs between what is allowed on one end and what is not allowed on the other.

We studied the likely scenarios resulting from security interoperability and developed methods that facilitate connectivity for grid application components despite such restrictions. We refer to our project as **Remus**, in which we are building a rerouting and multiplexing system that enables intra-firewall connectivity while minimizing the need to modify or rebuild either the grid applications or the firewalls. Our solutions are built on a number of standard secure communication protocols such as SSH and SOCKS. We present our experiments here and the results of tests using Globus and the Nimrod/G middleware.

We considered three types of conflicts that can hinder communications between two grid application components. A *node conflict* occurs when security restrictions block connections to the target node at any port. A *port conflict* occurs when security restrictions block connections to one or more target ports only. We may also find that, while a port conflict exists, there is instead another vacant port available. In such situation, the connection still cannot be made to the intended component until that vacant port is bound to some mechanism that will facilitate communication. We can also encounter a *protocol conflict*, when a successful connection can be made to the node at a given port but the service listening behind that port is unable to communicate using the protocol expected by the connecting component. We developed solutions that can handle each of those types of conflicts using various tested means through tunneling, port forwarding and communicating by proxy. **Remus** is also designed to accommodate, as much as possible, the network connections exactly as expected by the grid application components. Furthermore, it is recommended that rerouters use deliberately only those channels that are authorized by existing security policies.

We tested the applicability of **Remus** to Globus [5][6] and Nimrod [1]. Both are middleware that sit above and harness the computational power of several loosely-coupled processors. While Globus provides a generic facility for resource sharing and distributed computations, Nimrod specifically uses distributed resources for the computations of *parametric applications* [1]. Nimrod/G [2] is the Grid-enabled version of Nimrod that uses various grid-middleware services that support various standards, such as Globus [5]. Nimrod/G computations are not the same as Globus computations because Nimrod/G uses agents. When Nimrod/G uses remote resources, agents execute in them, allowing Nimrod/G greater flexibility in job management. Nimrod agents coordinate with Nimrod/G server components in order to perform data transfers to download input data or to upload output data. Nimrod agents may

also remain in memory and may be managed by Nimrod/G schedulers as the need arises, e.g., to abort jobs or be assigned new jobs.

Both Globus and Nimrod/G can get easily crippled by firewalls. A firewall can block the Globus gatekeeper from its client, e.g., Nimrod/G. A firewall can also block the computational nodes from a Globus job manager [6], or the Nimrod agents from the Nimrod/G server components. In our experiments, we were able to facilitate Globus job or data transactions and Nimrod/G experiments across two autonomous institutions where the firewalls were restrictive in both directions. One institution hosts Nimrod/G and accepts experiments, while the other institution hosts Globus and the computational resources for the experimental VO setup. We provided basic connectivity using SSH to tunnel connections through, with SSH being the open channel authorized by both institutions. In order to transparently facilitate component communications, we used SOCKS proxy tunneled through SSH. To minimize the need to customize or modify the network configuration used by the grid components, we used SOCKS wrappers that automatically reroute packets and send them across the SSH tunnel. Our experiments were successful with minimal installations and customizations required for SOCKS components and the facilitation of wrapped connections. There was no need to modify any piece of code for both Globus and Nimrod/G, nor was there a need to modify the firewall settings. Even the libraries used for SOCKS communication were dynamically linked, hence hard relinking was unnecessary.

In the near future, we will investigate the performance implications and potential bottleneck effects when Grid transactions are rerouted and tunneled in the manner we developed. We will also explore other means of getting across. Finally, we will continue to improve the internal architecture of **Remus** rerouters to make it easier for them to adapt to whatever deployment scenario they encounter, either through semi-automation features or through simple configuration directives.

References

[1] D. Abramson, R. Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parametised simulations using distributed workstations. In Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing, Virginia, August 1995.

[2] R. Buyya, D. Abramson, and J. Giddy. Nimrod/g: An architecture of a resource management and scheduling system in a global computational grid. In Proceedings of HPC Asia 2000, Beijing, China, May 14-17 2000.

[3] Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In Proc. 5th ACM Conference on Computer and Communications Security Conference, pages 83–92, 1998.

[4] Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: enabling scalable virtual organizations. International Journal of Supercomputer Applications, 15(3), 2001.

[5] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing, 11(2):115–128, Summer 1997.

[6] V. Welch. Globus toolkit firewall requirements. URL:http://www.globus.org/security/firewalls/Globuspdf.