Self-Management GRID Services – A Programmable Network Approach

L. Cheng[1], A. Galis[1], A. Savanović [2], B.J. Blažič [2], and J. Bešter[3]

[1] University College London, Electrical Engineering Dept., Torrington Pl., London, U.K.
{l.cheng,a.galis}@ee.ucl.ac.uk)
[2] Jozef Stefan Institute, Laboratory for Open Systems and Networks,
Jamova 39, 1000 Ljubljana, Slovenia
{arso,borka}@e5.ijs.si
[3] University of Ljubljana, Laboratory for Telecommunications,
Trzaska 25, 1000 Ljubljana, Slovenia
besterji@fe.uni-lj.si

Abstract. Due to the complexity and size of service oriented GRIDs, it is essential that GRID systems should be autonomous i.e. a self-management system is needed. This paper identifies the requirements of such a self-management GRID system and the required supporting services. This paper suggests that these supporting services should be deployed in the form of software modules through programmable techniques. This paper presents a communication protocol for dynamic self-configuration in programmable GRIDs as an example for supporting new network services.

1 Introduction to GRID

The concept of GRID networks [10], [11] was coined to enable transparent coupling of geographically-dispersed resources – such as computers, networks, remote data storage, scientific devices and instruments - for large-scale distributed applications in a highly distributed network environment, such as the Internet [1], [3]. There have been attempts to integrate GRID technologies with existing Web technologies by the Open Grid Service Architecture (OGSA) group [21], [24-28]. A major feature of the OGSA is that it is service-oriented, and the process of data exchange within a GRID should be transparent, efficient, and secure [2], [5], [7], [8], [9], [23]. The OGSA has defined three groups of services that occur within a wide variety of GRID systems [27]: *core services*, *data / information related services*, and *computation management related services*. This paper focuses on the last aspect. OGSA has also identified a range of supporting services, including data access agreement, resource provisioning and resource reservation, for supporting the program execution and resource management services in GRIDs [27]. Given the size of a Global GRID, manual management is not feasible. This paper suggests that there is a need for an *efficient*, *scalable*, *secured* [12] *self-managed* management system that is capable of dynamically compositing, installing, controlling and managing these supporting services; in turn these supporting services should provide services to the OGSA defined program execution and resource management services.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 141–148, 2004.

2 Self-Management in GRID

Self-management is defined as using autonomic principles to provide management functionalities. An important corollary is that the management function itself is an autonomic service, and is therefore subjected to the same requirements as other autonomic services. Self-management takes many forms [4]. Some fundamental building blocks include: a) self-configuration – this involves policies that direct changes to the configuration of a component, maintain desired functionalities or provide new functionality; b) self-healing – the system can detect, diagnose, and repair hardware, software, and firmware problems; c) self-protection, self-securing – the system can automatically defend against attacks, as well as detect and stop cascading internal failures; d) self-optimisation – the system and its components can continually seek ways to improve their behaviour and becoming more efficient; e) self-learning – the system learns from past events and morphs active policies to improve behaviour. Some of these building blocks combine into more powerful functions. For example, self-maintenance can be viewed as the combination of all of the above. The following are suggested supporting self-management services that should be deployed in a self-management system in a GRID environment [6]: a) monitoring of autonomic resources and workloads, for the detection of network operation glitches, monitoring dynamic network status, collecting required network information, locating resources and ensuring their availability, real-time memory management etc; b) intelligent decision-making facility - based on real-time network status, an intelligent module should decide when to switch tasks in order to utilise resources efficiently; c) dynamic reconfiguration of network entities' behaviour: automatically re-configuration of network entities' status to ensure real-time on-demand requests of users' tasks are satisfied; d) general system administrative functions: automatic system activity logging for administrators to debug; e) security services: entities on heterogeneous networks of a GRID should create security associations between each other. An autonomous decision-making facility should also decide whether to accept a service request based on pre-defined security policy. Note that in order to provide these supporting services, additional software modules should be installed on existing network devices. We shall discuss shortly how the dynamic composition, installation and management of these software modules could be achieved through programmable techniques

3 Programmable Techniques for Self-Management

3.1 Programmable GRID

As discussed previously, GRID computing requires distributed resources to be adaptively congregated and used based on service's needs, and a GRID management system should be self-managed due to the size and complexity of GRID. We have already outlined the requirements and the required supporting services that are needed for a GRID self-management system. Supporting services should be introduced to the current network in form of software modules. One approach to self-management is the creation of a programmable infrastructure [13] of computing and storage resources

that are dynamically reconfigurable during the execution lifetime of user applications; the dynamic feature of a programmable infrastructure enables the composition, installation and configuration of the service components that are required by the GRID self-management system. This is because programmable technologies [13-20] have been developed as an efficient and flexible tool for dynamically deploying new services and functionalities. With programmable technologies, the traditional store-and-forward network is transformed into a store-compute-and-forward type network: in the latter type of network, packets are no longer simple data packets but are now control packets that carry control information. Control information carried in a packet will be executed on programmable network nodes for node reconfiguration purposes and to meet service demands. New services in form of software modules and new network node functionalities may be installed and controlled on-demand. Thus programmable techniques provide the opportunity for dynamic loading and installation of software modules that provide and support the necessary self-management services in GRID. In the remainder of the paper an example self-management service is discussed, which is based on programmable techniques.

3.2 Overlay Autoconfiguration in GRIDs

A programmable GRID infrastructure should be able to operate as autonomously as possible in order to simplify management and facilitate scalability of these dynamic and potentially extremely complex systems. This requires developing a set of mechanisms and protocols for automation of different tasks in the management of programmable GRIDs. One such task is the automatic configuration of (core) overlays in programmable GRIDs, which is addressed in this paper.

Programmable GRIDs are network aware in the sense that they include active network nodes from the underlying communications infrastructure and explicitly use their processing and caching capabilities. This network awareness makes it possible to achieve performance gains for GRID applications simply by configuring the programmable GRID topology to match the underlying network infrastructure topology, i.e. directly connecting only neighbour active network nodes (ANNs) in the programmable GRID. On one hand, this enhances the performance of programmable GRID applications, because communications paths between GRID nodes are shorter and consequently communication delays are smaller. On the other hand, this also preserves the bandwidth in the underlying network, because information exchanged between GRID nodes is transfered along shorter paths and traverses each network path only once. With non-matching GRID and network topologies, data between GRID nodes can traverse the same network path more than once. Just as an example, consider a star GRID topology on top of the ring network topology. Last but not least, application of some active network facilities to programmable GRID systems, e.g. solutions for security [30] and interoperability [29] issues, requires that an ANN keep an up to date list of neighbour ANNs. Generally, neighbour ANNs in an programmable GRID overlay are generally not directly connected, i.e. they may be several hops apart. This makes neighbour discovery in an programmable GRID overlay a non-trivial problem and fundamentally different from physical neighbour discovery, which is based on a simple principle of boradcasting a query message on the physical link.

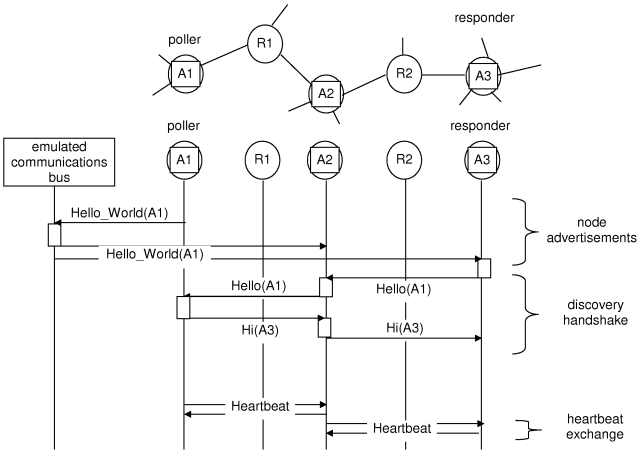

Fig.1. Protocol operation overview

3.3 Neighbour Discovery Protocol

The neighbourhood relation between a pair of ANNs in a programmable GRID is defined by the underlying network. Two nodes A1 and A2 are neighbours in a GRID overlay if network routes are defined such that there are no other ANNs on the path that packets traverse between A1 and A2. We have developed a protocol for automatic discovery of neighbour ANNs [36], which has three operation modes: node advertisements, discovery handshake, and heartbeat exchange.

The protocol uses four messages. The message header is common to all protocol messages, while messages differ in the format and semantics of the message body. Only a short description of the most important header fields is given here.

Message Type (MT, 4 bits): Specifies the type of the protocol message.
Intermediary (I, 1 bit): The main purpose of this flag is to trigger the discovery handshake, if set in the Heartbeat message.
Overlay ID (32/128 bits): Identifier of the programmable GRID overlay.
Source ID (32/128 bits): Identifier of the source node (see next paragraph).

Nodes use a single, globally unique ID to identify themselves to other nodes. For example, much like in the OSPF routing protocol [31], a node can select its ID to be the lowest or the highest among its IP-addresses. The format of all node ID fields in the message is determined by the value of the Address Type (AT) field in the header. Currently, the message can carry either a 32-bit IPv4 address or a 128-bit IPv6 address, which is defined by the value of the header field AT.

ANNs advertise themselves by periodically transmitting the Hello World protocol message to an emulated communications bus, which distributes the advertisement to all overlay members. The implementation of an emulated communications bus can be multicast-based [32, 33], with one dedicated multicast group per overlay, server based, or a combination of both. The Hello World message has an empty message body and the timeout value TO_{HW} defines the length of the period between the transmission of two consecutive messages Hello World by a node.

When the Hello World message is received, the receiving node (say, node X) only starts a discovery handshake with the sending node (say, node Y), if it does not already keep state for that node, i.e. keep it in neighbour list or keep state on a running handshake with Y. The handshake is started with probability *p*, which is used to limit the load on the node Y: if there are *N* members of a programmable GRID overlay, then on average only *(p*N)* nodes instead of all *N* nodes start a discovery handshake with sender Y simultaneously. Note that this does not mean that neighbour ANNs are chosen in a random fashion.

The discovery handshake is the main protocol mechanism for neighbour discovery and detection of network changes. It involves an exchange of two messages, Hello and Hi, between the node, which initiates the handshake (poller), and the node, which responds (responder). Both messages are processed and modified by every ANN en route between the poller and the responder, i.e. neighbour discovery is based on *packet interception* (see later on section).

The Hello and Hi messages contain two additional fields in the message body: the Destination ID and the Hop ID. Destination ID is the identifier (address) of the responder. Hop ID is the identifier of the current ANN en route between the initiator and responder. The transmission of the Hello message is always triggered, either by the message Hello World or by the I flag in the Heartbeat message. Similarly, the Hi message is sent only by the responder, indicated by the Destination ID in the Hello message, and only in response to the Hello message.

Upon interception of the Hello message an intermediate node first adds the previous hop ANN (Hop ID) to its neighbour list, unless it is already there. Then the node replaces the Hop ID value with its own identifier and forwards the Hello message towards the responder. The responder also adds the previous hop ANN into neighbour list and sends to the poller the Hi message, which is processed en route in the same manner.

The discovery handshake uses checksums, timeouts, retransmissions, and implicit acknowledgments to provide reliable message delivery service best suited to its needs. Use of TCP for this purpose is inappropriate for two reasons: its overhead is unacceptably high and its reliability service is not well suited for discovery handshake.

A node periodically sends the Heartbeat message to each of its known neighbours to indicate that it is still alive. The message body contains one additional field. the address of the destination node (Destination ID). This message is also processed by all ANNs en route to destination. ANNs use this message to refresh soft states in their neighbour list, detect changes in the network, and trigger a discovery handshake, when change is detected.

A node knows that it is intermediary, if its ID is not equal to the Destination ID in the Heartbeat message. If two nodes exchange Heartbeats, then they are neighbours and there should be no intermediate ANNs between them. Therefore an intermediate ANN sets the I flag in Heartbeat, thus indicating that the network change has occurred and that the receiver should start a discovery handshake.

We have seen that both discovery handshake and heartbeat exchange are based on packet interception. That is, an intermediate ANN en route between the poller X and the responder Y must intercept and process the protocol packet (message), even though it is not addressed to this intermediate ANN.

Contrary to traditional GRIDs, which are based on traditional IP networks, packet interception [34], [35] is inherently supported in programmable GRIDs, since ANNs

embody the flexible *store-compute-an-forward* model, which is in contrast to the *store-and-forward* model of traditional IP routers. Thus, by definition, ANNs in programmable GRIDs can intercept packets destined for some other IP address, process them in a specific way (aside from routing), and then forward them towards their destination.

3.4 Proof-of-Concept Implementation and Initial Test Results

A prototype implementation of the protocol with the ANTS toolkit [37] is being developed. Each protocol message is implemented in the form of a distinct ANTS capsule with simple forwarding routine. The core of the protocol is implemented as a separate module, which operates on top of the ANTS system and provides features for discovery of neighbour ANNs. By varying different protocol parameters, such as timeout values, retransmit counters, and probability p, the protocol convergence delay is within the range of 30 - 40 seconds, and network losses are within the range of 2% - 5%. The observed protocol overhead corresponding to this convergence delay is within the range 0.5 - 1 kb/s per node for the test configuration. Given these results and that the above convergence delay is within our target value for initial experiments, we find that the protocol performance (i.e. the observed protocol convergence delay and the network losses) meets our requirements. The above target value for protocol convergence was chosen, because the convergence delay of current routing protocol implementations ranges from around 30 seconds for OSPF [38] to several minutes for BGP [39]. We find the comparison with the routing protocols interesting due to the following reason: as far as the authors are aware, no results on convergence delay are currently available for related work in the area of automatic discovery of neighbour ANNs (e.g. [40]). On the other hand, routing protocols provide analogous functionality to the neighbour discovery protocol: after a perturbation occurs in the network, the routing protocol has to accordingly reconfigure the physical network (i.e. routes in the network) as soon as possible in order to minimise the disruption in communication caused by the perturbation.

4 Conclusions

The highly distributed feature and flexibility of GRID services makes them subject to the limitations of distributed programming such as performance, scalability, security. Existing GRIDs tend to be homogeneous. As such, GRID services would require some sort of autonomous self-managed management components and systems. These self-management capabilities should include: self-configuration, self-optimisation, self-healing, self-securing, self-protecting, and self-learning. This paper has identified a set of supporting services that are required for a self-management system for a GRID, and has suggested that these supporting services must be installed dynamically in GRIDs. We have presented a protocol that enables ANNs in programmable GRIDs to automatically discover neighbouring ANNs. This automates the management of core overlays in programmable GRIDs, improves the performance of GRID applications by reducing communications delay, enables specific solutions from active and programmable networks to be applied to GRIDs, and preserves bandwidth

by constantly adapting the overlay to match the underlying network topology in face of changes in the network infrastructure.

Acknowledgement. This paper partly describes work in progress in the context of the EU IST project CONTEXT [22]. The IST programme is partially funded by the Commission of the European Union.

References

1. http://www.hoise.com/primeur/03/articles/monthly/UH-PR-02-03-7.html
2. http://wwws.sun.com/software/cover/2003-1104/
3. http://www.datasynapse.com/solutions/diff.html
4. Waldrop, M.: Autonomic Computing – The Technology of Self-Management, http://www.thefutureofcomputing.org/Autonom2.pdf
5. http://eu-grasp.net/english/dissemination/presentations/Valles_industry.pdf
6. IST Project proposal, "Programmable GRID Network Solutions and Prospects"
7. http://www.gridcomputingplanet.com/news/article.php/2226121
8. http://setiathome.ssl.berkeley.edu/
9. http://biogrid.icm.edu.pl/
10. http://www.pallas.com/e/products/unicore-pro/index.htm
11. Galis, A., Gelas, J., Lefevre, J., Yang, K.: Active Network Approach to GRID Management & Services, in Workshop on Innovative Solutions for Grid Computing - ICCS 2003 Conference, LNCS 2658, ISBN 3-540-40195-4, Melbourne, Australia, June 02-04, (2003) 1103-1113
12. Smith, R.E.: Authentication: From Passwords to Public Keys, Addison-Wesley, ISBN: 0-201-61599-1, 110-112
13. FAIN (Future Active IP Networks) www.ist-fain.org
14. SNAP (Safe and Nimble Active Packets) http://www.cis.upenn.edu/~dsl/SNAP/
15. FAIN Internal Report R25.2 "SNAP, ANEP, and Security", Nov 2002 http://www.ist-fain.org
16. DARPA Program: www.darpa.mil/ito/research/anets/projects.html
17. Open Signalling Working Group, http://www.comet.columbia.edu/opensig/.
18. Galis, A., Denazis, S., Brou, C., Klein, C. (ed): Programmable Networks for IP Service Deployment, ISBN 1-58053-745-6; pp450, contracted for publishing in Q1 2004 by Artech House Books, 46 Gillingham Street, London SW1V 1AH, UK; www.artechhouse.com (2004)
19. Cheng, L., Galis, A., Eaves, W., Gabrijelcic, D.,: Strong Authentication for Active Networks, SoftCom 2003- 7-10 October 2003, Split (2003)
20. Suzuki, T., Kitahara, C., Denazis, S., Cheng, L., Eaves, W., Galis, A., Becker, T., Gabrijelcic, D., Lazanakis, A., Karetsos, G. -"Dynamic Deployment & Configuration of Differentiated Services Using Active Networks"- IWAN2003 Conference, 10-12 December 2003, Kyoto (2003)
21. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The Physiology of the Grid – www.globus.org/research/papers/ogsa.pdf
22. CONTEXT Project WWW Server: http://ontext.upc.

23. Berman. F.,, Fox, G.C., Hay, A.J.G.: Grid Computing- Wiley 2003, ISBN 0-470-85319-0,
24. http://www.globus.org/ogsa/
25. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1, W3C, 2001, www.w3.org/TR/wsdl (2001)
26. Ballinger, K, Brittenham, P., Malhotra, A., Nagy, W, Pharies, S.: Web Services Inspection Language Specification (WS-Inpection) 1.0, IBM and Microsoft (2001) http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html
27. Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.0 BEA Systems, IBM, Microsoft (2002) http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnbiz2k2/html/bpel1-0.asp
28. Foster: The Open Grid Services Architecture, http://www.nesc.ac.uk/teams/OGSAreviewPeterKunszt.pdf (2003)
29. Keneth L. Calvert.: Architectural Framework for Active Networks version 1.1. Technical report, University of Kentucky (2001)
30. Savanovic, A., Gabrijelcic, D., Blazic, B.J., Karnouskos, S.: An Active Networks Security Architecture. Informatica (2002) 26(2):211.221
31. Moy, J.: OSPF Version 2. RFC 2328 (1998)
32. Kosiur, D.: IP Multicasting. John Wiley & Sons, Inc. (1998)
33. Maimour, M., Pham, C.D.: An Active Reliable Multicast Framework for the Grids. In International Conference on Computational Science (2002) 588-597
34. Lefeevre, L., et. al.: Active networking support for the Grid. Lecture Notes in Computer Science (2001) 2207:16.33
35. Mohamed, N.: Active Networks and Their Utilization in the Computational Grid. Technical Report TR03-04-01, Engineering University of Nebraska-Lincoln, Lincoln, NE 68588-0115 (2003)
36. Savanovic, A.: Automatic Discovery of Neighbour Active Network Nodes. Technical Report IJS DP-8725, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia, January (2003)
37. ANTS. http://www.cs.utah.edu/flux/janos/ants.html.
38. Basu, A., Riecke, J.G.: Stability Issues in OSPF Routing. In Proceedings of SIGCOMM (2001) 225-236
39. Labovitz, C., Ahuja, A., Bose, A., Jahanian, F.: Delayed Internet Routing Convergence. In Proceedings of SIGCOMM (2000) 175-187
40. Martin, S., Leduc, G.: RADAR: Ring-based Adaptive Discovery of Active neighbour Routers. In Proceedings of IWAN 2002 (2002) 62-73