Preliminary version; to appear in 3rd High Performance Computing Conference (HPCC-07), Houston, TX, Sept 26-28, 2007.

Dynamic System-wide Reconfiguration of Grid Deployments in Response to Intrusion Detections

Jonathan Rowanhill[1], Glenn Wasson[1], Zach Hill[1], Jim Basney[2], Yuliyan Kiryakov[1], John Knight[1], Anh Nguyen-Tuong[1], Andrew Grimshaw[1] and Marty Humphrey[1]

[1] Dept of Computer Science, University of Virginia, Charlottesville VA 22094
[2] National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign, IL USA

Abstract. As Grids become increasingly relied upon as critical infrastructure, it is imperative to ensure the highly-available and secure day-to-day operation of the Grid infrastructure. The current approach for Grid management is generally to have geographically-distributed system administrators contact each other by phone or email to debug Grid behavior and subsequently modify or reconfigure the deployed Grid software. For security-related events such as the required patching of vulnerable Grid software, this *ad hoc* process can take too much time, is error-prone and tedious, and thus is unlikely to completely solve the problems. In this paper, we present the application of the ANDREA management system to control Grid service functionality in near-real-time at scales of thousands of services with minimal human involvement. We show how ANDREA can be used to better ensure the security of the Grid: In experiments using 11,394 Globus Toolkit v4 deployments we show the performance of ANDREA for three increasingly-sophisticated reactions to an intruder detection: shutting down the entire Grid; incrementally eliminating Grid service for different classes of users; and issuing and applying a patch to the vulnerability exploited by the attacker. We believe that this work is an important first step toward automating the general day-to-day monitoring and reconfiguration of all aspects of Grid deployments.

1 Introduction

As Grids become increasingly relied upon as critical infrastructure, it is imperative to ensure the highly-available and secure day-to-day operation of the Grid infrastructure. However, as Grid system administrators and many Grid users know, the deployed Grid software remains quite challenging to debug and manage on a day-to-day basis: Grid services can fail for no apparent reason, Grid services can appear to be running but users are unable to actually utilize them, information services report seemingly incorrect information, new versions of Grid software can need to be deployed, etc. Whenever a Grid administrator at a site determines that some part of their local Grid services is not running according to plan, and it is not immediately solvable via some local action, Grid administrators must generally contact their peers at other sites via telephone or email to debug overall Grid behavior. Email and

This material is based upon work supported by the National Science Foundation under Grant No. 0426972 and through TeraGrid resources provided by NCSA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

telephone are also heavily utilized to disseminate the availability of new versions of deployed Grid software, including those updates that contain important security patches. For such critical security-related events, this *ad hoc* process can take too much time, is error-prone and tedious, and thus is unlikely to properly remedy the situation.

The broad challenge addressed in this paper is: How can a large collection of Grid services by dynamically managed in near-real-time to provide an acceptable level of functionality without significant human involvement? At the core of our approach is Willow [1], a large-scale feedback control architecture featuring sensors, actuators, and hierarchical control logic that runs side-by-side to the Grid and uniquely facilitates near-real-time monitoring, reconfiguration, and general management of the overall functionality of a Grid. Specifically, in this paper, we narrow our focus to the design and implementation of the integration of ANDREA [2] -- the "actuation" part of Willow -- to ensure the security of large-scale Globus Toolkit v4 deployments (GT4 [3]) in the presence of an intruder (essentially a person gaining unauthorized access to some service). *We do not pursue ways in which to determine that a Grid intrusion has occurred, rather we assume that one has occurred and subsequently need to holistically react to such a breach.* In simulated experiments involving 11,394 Globus Toolkit v4 deployments, run over 768 real nodes, we show the application and performance of ANDREA for three increasingly-sophisticated reactions: shutting down the entire Grid; incrementally eliminating Grid service for different classes of users; and issuing and applying a patch to the vulnerability exploited by the attacker. We believe that this work is an important first step toward automating the general day-to-day monitoring and reconfiguration of all aspects of Grid deployments.

2 The ANDREA Management Framework

Willow [1] is a comprehensive approach to management in critical distributed applications. Willow achieves its goals using a combination of (a) fault avoidance by disabling vulnerable network elements intentionally when a threat is detected or predicted, (b) fault elimination by replacing system software elements when faults are discovered, and (c) fault tolerance by reconfiguring the system if non-maskable damage occurs. The key is a reconfiguration mechanism that is combined with a general control structure in which network state is sensed, analyzed, and required changes actuated. ANDREA is the "actuation" component of Willow. Because the focus of this paper is on the integration and use of ANDREA to control Globus installations in the context of intrusion detection, we will not discuss Willow any further (a complete integration of Willow with Grid mechanisms is the subject of continuing research). That is, we do not discuss the use of Willow to *detect* the intruder in question here, rather we focus on the use of Willow (specifically ANDREA) to *react* and take corrective actions.

2.1 Distributed, Hierarchical Management Policy in ANDREA

ANDREA is a distributed, hierarchical collection of nodes that integrate at the lowest level with the controlled system, which is in this case the Globus-based Grid. The collection of ANDREA nodes exist in a loosely coupled configuration in that they are not aware of one another's names, locations, or transient characteristics. This is a key point for controlling truly large-scale systems where the cost of any node maintaining a precise and up-to-date view of the remainder of the system is prohibitive (or the task is impossible). An ANDREA node advertises a set of attributes that describe important state of the application node it wraps. This is similar to a management information base (MIB) in traditional management architecture.

ANDREA nodes interact with one another through delegation relationships. Each ANDREA node: (1) states a policy describing constraints on the attributes of other ANDREA nodes from which it will accept delegated tasks; and (2) describes constraints on the kinds of tasks it will accept. This is known as a *delegate policy*. Meanwhile, if an ANDREA node seeks to delegate a task, it states constraints on the attributes of possible delegates. This is called a *delegator policy*. A delegated task is given to all ANDREA nodes in the system and *only* those nodes for which, within a time window specified by the delegator: (1) the delegator's attributes and the tasks's attributes meet the requirements of the delegate's policy; and (2) the delegate's attributes meet the requirements of the delegator's policy. Details of the addressing language and underlying mechanism used by ANDREA, called *Selective Notification*, are provided in the work by Rowanhill et al. [4].

Once an ANDREA node has delegated a task to other ANDREA nodes, there exists a loosely coupled management relationship between the nodes, because the delegator need not know the name nor locations of its delegates. It merely uses the "handle" of the management relationship to multicast a communication to its delegates. Likewise, a delegate need not know the name nor location of its delegator, and it need not be aware of any other delegates. It can communicate with the delegator using the management relationship as a handle. In addition, temporal coupling between nodes is minimized. Delegates can join a relationship at any time during the specified time window and receive all relevant communications in the correct order upon joining. This management relationship is particularly advantageous for users (or other software components) that are "injecting work" into the ANDREA system.

2.2 Hierarchical Workflow in ANDREA

Management is affected in ANDREA by defining *workflows.* A workflow is a partially ordered graph of task nodes, and a task can define a child workflow within it. A task with an internal workflow represents a composite action. A leaf task is one that contains no sub-workflow. A leaf task may be an action, a constraint, a query, or a periodic query. Each leaf task of a workflow states an *intent*, a token from an enumerated language known to all ANDREA nodes. A task's intent can represent any delegated entity. A common semantic is that each intent represents a requirement on distributed system service state defined as a *constraint* on system variables. Each workflow defines a *reason*, a token from a global enumerated language stating the purpose for the intents of the workflow. Conflicts are detected between the intents of tasks on the same blackboard, and resolved through preemptive scheduling based on the priority of reasons.

An application component can introduce management requirements into a system by defining a local workflow to its ANDREA service interface. This workflow is managed by the ANDREA system so that tasks and constraints are carried out in the right order and delegated to appropriate nodes. Each ANDREA node has a conventional hierarchical workflow and blackboard model.

It is impractical (and non-scaleable) for any single ANDREA node to keep track of all nodes in the system and thus be able to precisely determine *which* nodes have or have not complied with the intended action of a given workflow. Instead, delegates in a workflow keep *aggregate* determinations, in the form of histogram state. When a delegated task is completed (based on this aggregate state), the delegator rescinds the delegation. The management relationship that had linked the delegator to the delegates is terminated and eliminated and thus again fully decoupled.

2.3 Programming ANDREA

ANDREA has a unique programming model in which system configuration tasks (which are part of the workflows) are represented as *constraints*. The set of managed resources to which a constraint applies is described by a set of properties of the resource. Each system administrator (or automated system administration component) can issue "constraint tasks", i.e. system configuration workloads that put the system in a state with certain properties. ANDREA will configure the system and attempt to maintain the given constraints. Conflicting constraints are typically resolved by a priority scheme among administrators. For example, suppose an administrator issued the constraint that no low priority jobs should be run on a set of Grid nodes. ANDREA will reconfigure the selected nodes, and then continue to make sure this constraint is upheld. So, if another administrator issued a command that no medium priority jobs should be run on an overlapping set of nodes, ANDREA can configure the system to maintain both constraints (i.e., no low or medium priority jobs). However, if a command to allow low priority jobs is received, there is a conflict since both constraints cannot be simultaneously met.

3 Using ANDREA to Reconfigure the Grid in Response to Intrusion Detections

While we believe ANDREA can be used for many aspects of day-to-day Grid management, our focus in this effort is to use ANDREA to reconfigure the Grid in response to intrusion detections. We generically describe the scenario as follows: an operator has manually discovered a rootkit on a node in the Grid and relays this information to the Grid control center, which uses ANDREA to put the system into a number of different configurations on a per-node basis[1]:

[1] In the future, we will use the sensing capabilities of Willow to discover such rootkits and automatically engage ANDREA (with human confirmations as warranted).

- *Normal* – nodes are accessible to all user classes
- *Shutdown* – no users are allowed to access nodes (since they are considered infected)
- *No low priority* – no low priority users are allowed access to nodes. Eliminating this class of resource consumption reduces the dynamics of the Grid so that it is easier to assess the extent of the intrusion, without shutting down the entire Grid.
- *Only high priority* – no low or medium priority users are allowed access to nodes. If "*no low priority"* is not sufficient, this configuration further simplifies the Grid environment while still offering some level of service.
- *Patch nodes* – in the case that the rootkit is exploiting a known vulnerability, grid system administrators can use ANDREA to take nodes identified as vulnerable offline, patch them, and bring them back online. This will cause a percentage of the Grid nodes to be unavailable for a time, but will allow other nodes (ones without the vulnerability that allowed the break in) to remain operational.

While this may seem like a simple scenario (and a simple "shutdown" solution), it is an ideal illustrator of the tasks for which ANDREA was designed. Every node in the system must be informed of the vulnerability and (as we shall see) ANDREA can provide a probabilistic description of "the grid's" response (i.e. "the grid is now 90% patched"). While an equivalent effect of avoiding the vulnerable nodes may be achieved via the manipulation of scheduling policies in either a single global queue or via coordination of scheduling policies on multiple queues in the Grid, we believe that this is not a reasonable approach for a number of reasons. First, it is extremely unlikely that there is a single queue, and coordination of multiple queues requires sites to give up site autonomy, which is unlikely (in contrast, as will be discussed in Section 5, ANDREA can be configured to respect site autonomy). Second, we require a mechanism affecting all Grid access, not just access to a job submission queue. Third, ANDREA is part of the larger Willow functionality, which will in the longer term automate some of the sensing and control-rule aspects implied here. Solely manipulating queuing policies offers no automated solution for integrating the discovery of a rootkit and an appropriate reaction.

In the remainder of this section, in the context of intrusion detection and reaction, we describe how to deploy ANDREA for the Grid (Section 3.1), the Grid-specific ANDREA workflows (Section 3.2), and the actual mechanisms executed on the controlled nodes to reconfigure the Globus Toolkit v4 Grid (Section 3.3).

3.1 Deployment of ANDREA for the Grid

An ANDREA deployment consists of a set of ANDREA nodes, which are processes that can receive configuration messages, enforce constraints, and report status. It is up to the ANDREA system deployer to determine how many managed resources are controlled by an ANDREA node, but typical deployments have one ANDREA node per machine. In the experiments reported in this paper, each deployment of GT4 on a machine was managed by a single ANDREA node. Each ANDREA node uses an XML deployment descriptor for the various supported system configurations and the possible conflicts between those configurations. (The mechanism by which ANDREA maps these labels into actual operations on the Grid is discussed in Section 3.3.) From this XML description of the supported configurations, ANDREA can infer that “patch nodes” mode conflicts with the “shutdown”, “no low priority” and “only high priority” modes since they are subsets of “normal” mode. Configuration tasks, i.e. “go into normal mode”, are issued with an attached priority that is used to resolve conflicts. In this way, configuration tasks can be dynamically prioritized. The current security model in the ANDREA implementation still requires receiver-side authentication and authorization checks to fully ensure that a received command is legitimate.

3.2 Using ANDREA Workflows for Configuring the Grid

System administrators (and potentially automated entities) inject workflows into their ANDREA nodes to manage and configure the Grid. Some of the tasks in a given workflow are delegated from the controller to all ANDREA nodes with particular attributes and acceptance of the controller’s attributes, as described previously. The result is the arrival of a task at a set of ANDREA nodes managing Grid components, with the members of the set based on the policies stated at controllers and managers. Once an ANDREA node receives a task, it carries it out by actuation of the managed Grid components.

The status of the action, constraint, query, or periodic query is periodically sent back to the task’s delegator on the order of seconds. ANDREA’s distributed algorithms perform aggregation of this status information from all other delegates of the task. The command issuer can use these status updates to make determinations about the overall system state based on the percentage of nodes that have responded. ANDREA includes mechanisms to map task status histograms to representation of the task’s state at the commander. For example, the Grid may be considered to be in a certain configuration when 90% of the nodes have replied that they are in that configuration, as 100% responses may be unlikely or unnecessary in large systems, depending on the situation.

Once a task is received by an ANDREA node, it must decide what action to take. The actual action an ANDREA node takes depends on what other active tasks are already being enforced. For example, if an ANDREA node is enforcing a “normal” constraint and a task arrives to enforce the “patch nodes” constraint, the net effect will depend on the priority of the “patch” command. If it has a higher priority then it will be enforced in place of the “normal” configuration. However, if it has a lower priority, it will not be discarded. Instead the “patch” command will remain on the ANDREA node’s blackboard. If the “normal” configuration is withdrawn by its issuer, then the highest priority commands still on the blackboard that do not conflict with one another will be the ones that are enforced.

3.3 Integration of ANDREA with GT4 (A2GT4)

Although each local ANDREA manager knows which constraints apply to the resources it manages, it requires a resource- (or application-) specific mechanism to enforce those constraints. In our system, enforcement is performed by a system component called ANDREA-to-GT4 (A2GT4). A2GT4 is the mechanism by which

ANDREA can alter the behavior of Globus Toolkit v4 and also includes an actuator, plugged into each ANDREA node's blackboard, defining how and when to run the mechanisms based on tasks on the blackboard and their scheduling.

We developed A2GT4 based on particular scenarios. Each new scenario is additive to the existing A2GT4 mechanism (note that ANDREA itself handles conflicts between concurrent re-configurations). With regard to the specific intrusion-detection scenario, this means that we had to map each of the five identified states ("normal", "shutdown", "no low priority", "only high priority", and "patch nodes") into a particular configuration of the GT4 services. In these experiments, the A2GT4 mechanisms manipulate authorization mechanisms to control the overall performance of a GT4 services. We chose this approach because the scripts were easy to debug, easy to confirm correctness, and not prohibitive of more sophisticated approaches necessary for different scenarios (which are under development).

4 Experiments

To evaluate the performance of our Grid management methodology, we deployed 15 instances of the Globus Toolkit, each managed by a local ANDREA node, on 768 nodes of the NCSA IA-64 TeraGrid cluster. Each cluster node had two 1.3-1.5 GHz Intel Itanium 2 processors and 4-12 GB of memory, with a switched Gigabit Ethernet interconnect. Excluding approximately 1.1% of the instances that failed to correctly start up for various reasons, this single NCSA cluster allowed us to simulate a much larger Grid of 11,394 ANDREA nodes and Globus Toolkit instances (Note that it was a "simulation" because it *wasn't* an actual wide-area Grid deployment, although we truly had 11,394 independently-configurable Globus deployments in our experiments). 101 of the computers each ran a Selective Notification node, forming an overlay network in the form of a tree with a branching factor of 100. Each of the ANDREA nodes was connected as a leaf to this overlay network, such that each lower level Selective Notification dispatcher supported roughly 115 ANDREA nodes. The ANDREA controller ran on a machine at the University of Virginia.

We performed three experiments using the attacker/intruder scenario introduced in Section 3, with normal, no low priority, only high priority, and patch node configurations. For each experiment, we issued configuration commands from the ANDREA controller at UVA and logged the results at UVA and NCSA. At UVA, ANDREA logged the times at which commands were issued to and acknowledged by the ANDREA nodes at NCSA. At NCSA, the A2GT4 scripts logged each state change to the cluster's file system. Over 200,000 events were recorded.

4.1 Experiment 1: Shutdown

The first experiment evaluated the ability of the ANDREA manager to shutdown all Grid nodes in the event of an attack or break-in. An ANDREA controller issues the command for all nodes to enter the shutdown state, prompting all ANDREA nodes to run the A2GT4 shutdown script, which denies access to all Globus Toolkit services on that node.

In Figure 1, we see that within seconds of issuance of the Shutdown command from UVA, the A2GT4 shutdown scripts begin running at NCSA, with all nodes shutdown 26 seconds after the command was issued. ANDREA confirms the state change of most nodes after 16 seconds, and all 11,394 nodes are confirmed shutdown after 26 seconds.

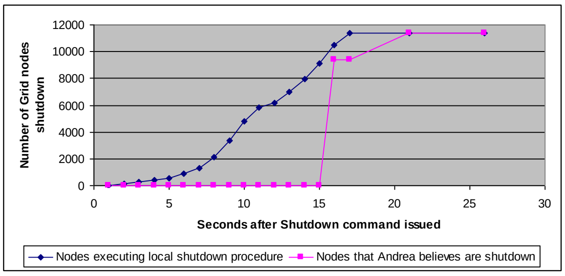

Figure 1: Testing Basic ANDREA Control of GT4 Grid Nodes

4.2 Experiment 2: Shutting out User Groups/Classes to Enable Better Damage Assessment

The second experiment evaluates the ability of ANDREA state changes to control Grid jobs. We periodically submitted batches of Globus WS-GRAM jobs using three different user credentials, where the users and their jobs are prioritized as low, medium, and high. As the Grid comes under attack, an ANDREA controller decides to limit access to medium and high priority users only, to reduce the number of accounts to monitor while continuing to serve higher priority customers. As the intruder/attack continues, a separate ANDREA controller decides to limit access to only high priority users to further lock down the system.

Figure 2 shows the number of high, medium, and low priority jobs running during the experiment. The number of running jobs follows a cyclical pattern, as a new batch of jobs is submitted while the previous batch is completing. The first ANDREA controller issued the command to deny access to low priority jobs 80 seconds into the experiment. Each ANDREA node ran the A2GT4 *NoLow* script, killing all running low priority jobs and updating the Globus authorization mechanism list to deny access to low priority users. 24 seconds later, all low priority jobs had stopped running and ANDREA acknowledged the change to the no low priority state. While the low priority user continued to attempt job submissions, they were all denied due to the access control change. At 275 seconds into the experiment, the ANDREA manager issued the command to deny access to both low and medium priority jobs. ANDREA ran the A2GT4 *HighOnly* script, killing all running medium priority jobs and updating the Globus access control list to allow access to only high

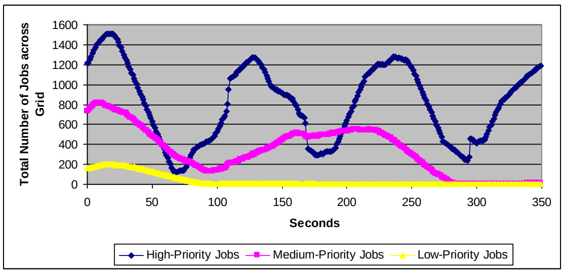

Figure 2: Incrementally Eliminating Classes of Users

priority users. 16 seconds later, all medium priority jobs had stopped running and ANDREA acknowledged the change to the only high priority state. These results demonstrate how ANDREA efficiently propagated state changes initiated by the manager at UVA across the 11,394 nodes at NCSA and the A2GT4 scripts effectively reconfigured the Grid environment on the nodes.

4.3 Experiment 3: Patching Vulnerable Nodes

The first two experiments evaluated the functionality and performance when the ANDREA manager issued constraints defining new states that the nodes should maintain, such as shutdown, no low priority, or only high priority. The third experiment evaluates ANDREA actions, which are commands that are executed once per node, and also demonstrates the use of ANDREA's selective notification.

We artificially configured 20% of the ANDREA nodes to report their operating system to be RedHat 7.3, while the other nodes report other operating systems. In this scenario, we assume that the operator determines that the specific compromise arose as a result of a vulnerability in Red Hat 7.3, and the operator want to use ANDREA to issue an instruction *to only those vulnerable* nodes to take themselves offline, apply the patch, and then re-register with the Grid (i.e., come back on-line). Figure 3 illustrates that it took about 7 seconds for all 20% of the nodes (approx. 2250 nodes) to initiate the *patch* command within A2GT4. We simulated that the operation of retrieving the patch (the patch itself is not contained in the control signals provided by ANDREA, although a URL could be included), installing the patch, and then rebooting (which subsequently brings the Grid node back on-line) took 3 minutes in its entirety. Hence, these patched Grid nodes began appearing back on-line at 180 seconds into the experiments; all nodes were back on-line and reported to ANDREA within 13 seconds after that. While not directly shown in Figure 6, ANDREA's Selective Notification facilitated the "addressing" of the *Patch* command to only those nodes that published the property of being Red Hat 7.3 -- the actual destination (IP Address, port) of relevant nodes was handled by ANDREA. Only simulated Red Hat 7.3 nodes received such a command.

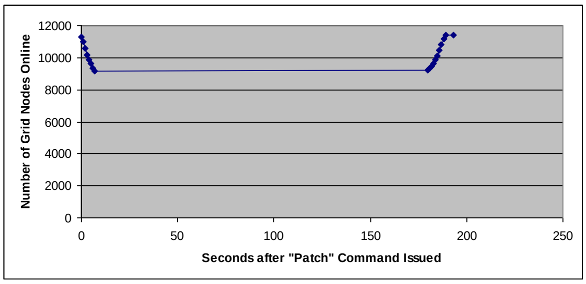

Figure 3: Applying a Patch to Only Red Hat 7.3 Nodes

5 Discussion

We believe that our experiments uniquely show a novel approach to the semi-real-time management of large-scale Grids. However, some discussion of the results is warranted. First, there is an obvious question of the base-line against which these results can be compared. Unfortunately, we are aware of no Grid systems that actuate changes on system components in the automated manner at the scale of which ANDREA is capable. Intuitively, it seems that being able to issue re-configuration commands (such as “shutdown” or any other command) to over 11,000 receivers in ~25s seems not unreasonable for many application domains (while our experiments take place in a simulated environment of millisecond-latency between nodes, we believe that in a real wide-area deployment, network propagation delays will not have an overwhelming effect on these durations, as the dominant effect is based on the algorithms and topology of the ANDREA control structure, hierarchy, and fan-out as used in these experiments).

Second, there is the question of the cost of ANDREA’s ability to scale. In other words, ANDREA’s ability to reach large numbers of nodes, addressed by properties, and aggregate responses from those nodes may make it inappropriate for smaller scale communications tasks. We are not claiming that ANDREA should be used as a replacement for current grid communication mechanisms. Instead, we wish to use ANDREA for the task of managing large-scale distributed systems such as Grids and ANDREA’s performance should be viewed in this context. So, while ANDREA would undoubtedly be slow when compared to a single grid communication (these days typically a web service call), its performance, again intuitively, seems reasonable when compared to 11,394 such calls.

In general, we recognize that there are clearly a number of issues that must be addressed before our approach is ready for an operational Grid like the TeraGrid:

- The scope of the A2GT4 mechanism must be expanded to all functionality of the Grid node beyond those capabilities provided by GT4 (e.g., SSHD).
- A more comprehensive mechanism to put the managed node into each particular state must be developed. For example, we realize that the "shutdown" mechanism we implemented is of negligible value in the unlikely event that a vulnerability in the GRAM service allowed an attacker to *bypass* the GT4 authorization mechanism!
- A comprehensive suite of use-cases that cover the majority of day-to-day operations must be developed and reflected in A2GT4 and the ANDREA workflows. For example, we are currently investigating a scenario in which a high-importance operation that relies on the Grid (such as tornado prediction) is not meeting its soft deadlines. The corrective actions to take, particularly *at the same time* as an event such as an ANDREA reaction to intrusion detection, is the subject of on-going research in this project.
- The security of ANDREA must be further analyzed and improved. Certain operations in ANDREA are not mutually authenticated, instead relying on a trust model that in some cases oversimplifies the actual requirements of a TeraGrid-like environment. In addition to analyzing and correcting for vulnerabilities within ANDREA itself, we would like to leverage the authentication infrastructure of the Grid itself, namely GSI [16].
- The entire functionality of Willow (including the sensing and control logic) must be applied to a Grid setting, leveraging existing solutions. For example, regarding the sensing capability/requirement of Willow, we believe that existing tools such as INCA [5] could be leveraged as the foundation for such operations.

More broadly, in our experiments, for simplicity we had a single control center, which is not unlike the Operations center of the OSG or the TeraGrid. But Willow/ANDREA supports multiple "control loops", such as a "VO controller" and an "Enterprise controller”. Ultimately, the decision regarding what to enact (even in the presence of conflicting commands) lies with the controlled nodes. Willow/ANDREA's control mechanism accommodates sites *not* executing commands requested of them.

Finally, it is clear that ANDREA can be used to manage the Grid, but one can reasonably wonder about the management of ANDREA itself! Currently, the assumption is that ANDREA is running on a dedicated, reliable network of reliable machines. This is plausible for most situations but clearly not applicable for the broader Internet itself. It is the subject of future work to make ANDREA itself manageable.

6 Related Work

There exist many hierarchic management architectures in the literature, some commercial, others experimental. The size of the managed systems and the scope of management vary with architecture.

Applying the model of Martin-Flatin et al [6], we can describe management architectures by the nature of their delegation capabilities. Early management architectures, such as traditional SNMP based systems [7], apply centralized control. SNMP-based systems have evolved to apply management hierarchy. Examples of hierarchic management systems include the SAMPA [8] architecture, the Tivoli Management Framework’s [9], and Open Systems Interconnection (OSI) standard [6]. In each latter case, the power of late-bound, run-time delegation service is increased, allowing managers to delegate authority through commands over otherwise static infrastructure in static hierarchical configurations. These architectures assume capability divided into static, implicit hierarchy assigned at system design time.

Many of the tools of large-scale Grid deployments can be utilized in Willow/ANDREA. Existing monitoring systems, such as the Globus Toolkit MDS [11], the TeraGrid INCA system [5], and MonALISA [12], could provide sensor data to ANDREA for analysis. For software configuration management of Grid systems, ANDREA could interface with package managers such as Pacman [14] and GPT [15] to (for example) patch vulnerable software versions.

ANDREA’s delegation policies, matching tasks to nodes, are similar to Condor’s ClassAd Matchmaking [16], though we believe ANDREA’s mechanisms for resolving conflicting constraints are more powerful. The Hawkeye monitoring system [13] builds on Condor’s ClassAd capabilities.

Evolving grid/web service standards can also important for the management of large-scale grids. WS-Notification [18] and WS-Eventing [19] define message formats for the delivery of asynchronous messages. ANDREA’s selective notification system could be modified to send messages that are consistent with these standards. WS-Management [20] and WSDM [21] are both specifications that define a standard language used for management of web services. In the future, ANDREA represents a mechanism by which WS-Management (or WSDM) commands can be enacted and monitored on a wide scale.

7 Conclusion

We have argued in this paper that for security-related events such as the required patching of vulnerable Grid software, the *ad hoc* process of human operators using email and telephones can take too much time, is error-prone and tedious, and thus is unlikely to completely work. The Willow/ANDREA approach provides powerful mechanisms for dynamic reconfiguration of Grid nodes in the presence of intruder detections. We have demonstrated, through the development of A2GT4 and experiments involving 11,394 Globus Toolkit v4 deployments, that ANDREA can effectively enact a variety of different holistic Grid reconfigurations to limit the vulnerabilities. We believe that this work is an important first step toward automating the general day-to-day monitoring and reconfiguration of all aspects of Grid deployments.

References

[1] J. C. Knight, D. Heimbigner, A. Wolf, A. Carzaniga, J. Hill, P. Devanbu, and M. Gertz., "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications," Intrusion Tolerance Workshop, DSN-2002 The International Conference on Dependable Systems and Networks, June 2002.

[2] J. Rowanhill, and J.C. Knight. ANDREA: Implementing Survivability in Large Distributed Systems. University of Virginia technical report. April, 2005

[3] Foster, I. and Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications, 11* (2). 115-128. 1997.

[4] Rowanhill, Jonathan C., Philip E. Varner and John C. Knight. *Efficient Hierarchic Management For Reconfiguration of Networked Information Systems.* The International Conference on Dependable Systems and Networks (DSN 2004). Florence, Italy, June 2004.

[5] INCA Test Harness and Reporting Framework. http://inca.sdsc.edu/

[6] J. Martin-Flatin, S. Znaty, J. Hubaux. "A Survey of Distributed Network and Systems Management Paradigms" Journal of Network and Systems Management. 7(1). pp 9-22. 1999

[7] M. Mountzia, D. Benech. "Communication Requirements and Technologies for Multi-Agent Management Systems." Eighth IFIP/IEEE International Workshop on Distributed Systems Operations and Management (DSOM'97). Sydney, Australia. 1997.

[8] M. Endler. "The Design of SAMPA." The Second International Workshop on Services in Distributed and Networked Environments. June 1995. IEEE Press.

[9] S. D. Taylor. "Distributed System Management Architectures". Masters Thesis, 1006, Univeristity of Waterloo, Ontario, Canada.

[10] G. Valetto, G. Kaiser. "Using Process Technology to Control and Coordinate Software Adaptation." 25th International Conference on Software Engineering. Portland, Or. May, 2003.

[11] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Grid Information Services for Distributed Resource Sharing. *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10),* IEEE Press, August 2001.

[12] I. Legrand, H. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, and C. Dobre. MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications. *CHEP 2004,* Interlaken, Switzerland, September 2004

[13] Hawkeye: A Monitoring and Management Tool for Distributed Systems. http://www.cs.wisc.edu/condor/hawkeye/

[14] Pacman. http://physics.bu.edu/~youssef/pacman/

[15] GPT: Grid Packaging Tools. http://www.gridpackagingtools.org/

[16] R. Raman, M. Livny, and M. Solomon, Matchmaking: Distributed Resource Management for High Throughput Computing. *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing,* July 28-31, 1998, Chicago, IL.

[17] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. A National-Scale Authentication Infrastructure. IEEE Computer, 33(12):60-66, 2000.

[18] OASIS. Web Services Notification (WS-Notification) TC. http://www.oasis-open.org/ committees/tc_home.php?wg_abbrev=wsn

[19] D. Box *et. al.* Web Services Eventing (WS-Eventing), August 2004, http://ftpna2.bea.com/ pub/downloads/WS-Eventing.pdf

[20] R. McCollum, eds. Web Services Management (WS-Management), June 2005, http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-management.pdf.

[21] OASIS. Web Services Distributed Management TC. March 2005, http://www.oasis-open.org/ committees/tc_home.php?wg_abbrev=wsdm