Towards a Usage Monitor for Enforcement of Usage-Based Access Control in Grid Computing

Russ Wakefield
Department of Computer Science
Colorado State University
waker@cs.colostate.edu

ABSTRACT

Grid computing is evolving as a solution to resource issues associated with large-scale computing. As the technology to manage the resources of the grid evolves beyond multiple logins and Unix file permissions, access control becomes a viable solution as it has in many application spaces. Usage-based access control solutions resolve the need for attribute based resource security that can handle mutable conditions. Defining the access control model is not sufficient; the model must be enforced for the technology to be effective. This paper identifies the requirements for a systems monitor to enforce the mutability requirements of usage-based access control in a grid computing environment by examining both the existing grid environment and similar access control environments such as digital rights management (DRM). These requirements will be taken forward into an implementation.

Categories and Subject Descriptors
Not sure what goes here

General Terms
Theory, Security, Grid Computing

Keywords
Access Control, Enforcement, Monitor, Quality of Service, Virtual Organization, Virtual Environment, Authorization

1. INTRODUCTION

Grid computing is defined as coordinated resource sharing and problem solving within a dynamic, multi-institutional, virtual organization [4]. Originally, grid computing theory indicated that resources that were unused within an environment could be added to the greater good. This is not as true today, as dedicated grids now exist such as TeraGrid [18] and the Earth System Grid [19]. These grids create a *virtual supercomputer* to accomplish large-scale scientific analysis. The resources necessary for the virtual supercomputer number too great to be owned by a single entity or company – instead organizations with a common purpose band together to form the virtual organization to supply these resources. TeraGrid is available for open scientific research and boasts over 750 teraflops of computing capability and more than 30 petabytes of data storage. Earth System Grid II was created to enable climate analysis and knowledge development from global system models. Users belonging to these VOs coordinate usage of the virtual supercomputer through their VO administrative domain. The promise of massively parallel systems has been with us for years – but the implementation of this promise is just beginning to evolve within the grid computing environment.

Virtual Organizations are defined as an organizational entity made up of corporate, non-profit, educational, or other units that exist in a diverse geographic space and use computer networks to communicate for distributed computing. [28] Virtual Organizations (VOs)

[5] provide the coordination necessary to make the geographically-diverse heterogeneous resources owned by a multitude of institutions available to the members of the organization. This coordination involves the use of middleware packages to ensure the resources identified by the members of the virtual organization are made available in a manner that is timely as well as safely and securely.

Resources and services may be shared across multiple VOs. When the blueprints for the Open Science Grid [11] were created, the need to "contract" services and resources from a site participating in one VO to another was identified as a key component of the VO. A dataset key to climate research performed on one of the sites for the Earth Systems Grid and to the research of pollution effects performed on TeraGrid would cause that site to be a VO member of both grid environments. The control of this resource is relegated to the site and the contract is brokered between the two participant VOs by the site. In this situation, the site is allocating resources (compute cycles, data space) and services (access to the data set) to both grids. Access control systems in general must be able to handle this situation.

These resources are owned by the members of the virtual organization, and as such the members want to retain local control over their resources. This adds an additional dimension that traditional access control models have not had to handle. Current access control mechanisms within grid computing are similar to Unix file system controls, handling resource management by mapping local users to global users or defining a generic user account for the grid users to get their authorization from.

Usage-based access control is defined as the ability to allow/restrict the use of a specific resource by a specific entity based on identity and on changing conditions. These changing conditions are referred to as *mutable. Mutability* requirements are the key to the differences between traditional access control and usage-based access control. The ability to continue to have *decision points* during the subjects use of a resource is increasingly becoming important in such environments as DRM and collaborative computing. Park and Sandhu [13] have presented a usage-based access control model called $UCON_{ABC}$.

The $UCON_{ABC}$ model presents a definition of the mechanisms for handling mutability of attributes based on usage. While a usage monitor would not be tied to the $UCON_{ABC}$ model, the model does allow us to examine the associated definitions for requirements of the usage monitor.

Digital Rights Management is defined as access control mechanisms used by publishers, corporations, and copyright holders to limit the usage of digital media or devices [29]. Digital Rights Management (DRM) systems are access control systems that incorporate usage as a part of their model. Several of these systems have been fully implemented and are in use today. Enforcement is a key component of a fully implemented DRM system; we take a look at several of these systems to examine how they provide enforcement.

Section 2 takes a look at the existing grid environment, and at on-going projects to implement attribute-based access control within the grid environment. In particular the GribSheb projects and the VO Federation projects are examined for implementation details that would put requirements on a usage monitor. Section 3 examines the functions presented within the UCON model to identify the requirements of the local monitor to implement the mutability requirements of the model. Section 4 looks at existing DRM models to identify design and implementation issues. Section 5 summarizes the requirements gathered from the previous sections. Section 6 examines related work, and Section 7 looks at our conclusions and future work.

2. THE GRID ENVIRONMENT

The term grid computing was coined in the late 1990s to describe a set of resources distributed over wide-area networks that can support large-scale distributed applications [4]. The similarities between a vision of readily available

computer resources and the existing power grid boosted the analogy – the desire to be able to "plug" into the computational grid and have access to computation and data is likened to plugging an electrical device into an outlet and drawing power to operate it. The lure of being able to utilize these easily accessible resources has provided the focus for a tremendous amount of research and development. As more and more applications begin using the grid environments, data about the running the applications within that environment becomes available and the requirements to provide tools to aid in the use of the grid environments solidify and become clearer.

One of the biggest issues surrounding the grid environment is the complexity of the various components. There are potentially thousands of services, hardware components, nodes, software components, and administrative domains involved in the execution of an application within the grid. As with its predecessors in traditional computing, this has led to the development of all-encompassing packages to add layers of abstraction to reduce the complexity.

Foster et al [5] added the concept of a virtual organization to the definition of grid computing – defining the VO as a set of participants with various relationships that wish to share resources to perform a task. Resources are defined to include data, computers, scientific instruments, software, etc. The virtual organization is key to achieving quality of service, within this social framework occurs the Service Level Agreements (SLAs) necessary to commit these resources. According to *Anatomy*, "VOs enable disparate groups of organizations and/or individuals to share resources in a controlled manner, so that members may collaborate to achieve a shared goal". These virtual organizations have become a de facto standard used in discussing the grid and working through the social issues associated with using multiple owners of multiple resources each with its own set of policies.

Just like Internet protocols have layers enabling TCP and HTTP to use the common IP layer, providing a layered approach on the grid enables each of the layers to support different architectures and behaviors above it through the use of Applications Programming Interfaces (APIs). An open architecture enables many development approaches, the establishment of the common protocols and interfaces is essential to the success of implementing abstraction to work within the grid.

A typical layered Grid architecture and how it relates to the similar layered Internet protocol looks like this [5] [6]:

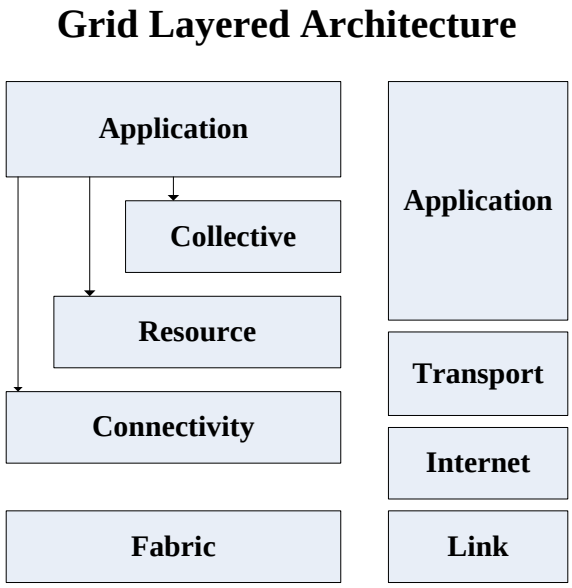

Figure 1: Comparison of Grid and Internet Protocol Layers

The fabric layer provides the nodes, services, software entities, etc. that are available for the applications to use and are mediated as a part of the grid protocol. The fabric components are responsible for implementing the operations the users of the resource need to accomplish their tasks. The connectivity layer is for communication and authentication in grid-specific transactions. Characteristics of authentication include single sign-on, delegation, and interoperability with resource-specific security, and cooperation within the virtual organization. The resource layer facilitates information gathering and control

functionality at the level of individual resources; it is strictly concerned at the single resource level. The collective layer provides the communication between the resources to provide interaction between them, such as a directory service or a data replication service. On top of this stack is the application layer made up of the applications that participants within the virtual organization want to run within the grid.

Virtualized environments have become common in strategies used to provide grid computing. These environments [8][9] provide a mechanism for the application to be run on a system while protecting the local resources on that system by "sandboxing" [8] the application. The platforms for these environments range from desktop to server and provide a wide range of flexibility.

These environments create a default account on the system for the grid applications to run under. This level of granularity does not lend itself to handling large numbers of users in a distributed computing system.

The advent of virtualized environments has led to the implementation of Service Oriented Architectures (SOA) in which the grid user acquires the resources necessary to accomplish their task through a third-party service provider, such as IBM [14]. This approach has a great deal of flexibility in a grid environment of application usage – the service provider keeps the applications (and licenses) up to date.

The basic building blocks of a VO are a diverse and homogeneous set of resources, and a large user base. These building blocks, working within the distributed nature of grid computing lead to one of the greatest challenges - the administration associated with the security of the grid.

The current de facto standard for security in grid computing is the security component within the Globus toolkit [12]. This component uses the "grid mapfile", an access control list mapping local and global identities. Once this mapping has occurred, the global identity is allowed to use resources on the host at the level defined by the local identity.

This approach lends itself to defining a "grid" account that everyone that comes in through the grid scheduling uses. Within the VO, there may be hundreds or thousands of users – the granularity desired for making access control decisions would be difficult using this mechanism.

Authentication becomes an issue in a grid environment that may not allow all grid users to have the same privileges. Simply authenticating the user to use the system is not adequate; each user may have different privileges based on not only identity, but on usage and system conditions. An example would be a company that allows grid users to purchase 75% of the processing cycles available during the evening, leaving the remaining 25% of the cycles available for its own data processing needs.

Grids are federated environments – each local site has to retain control of its own resources while allowing grid users to pass inspection. The Globus team is addressing this need through the development and integration with the **Shibboleth** project [15], adopting the notion of attribute-based authorization. According to their website, "The Shibboleth® System is a standard based, open source software package for web single sign-on across or within organizational boundaries. It allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner."

Shibboleth is a project of the Internet2 initiative, a research and development consortium of a set of universities, companies, and organizations focused on the development of next generation network applications within the research and education environment [16]. Although the goals of the Shibboleth are focused on the educational environment, many of them are directly applicable to the grid environment. These include:

- Multiple authorizations for multiple applications
- Account management
- Third party services authorization
- Interoperability
- Localized authorization technology
- SPs to control their own resources

While this implementation provides the ability to have attribute-based access control with a single sign-on, it does not address the ability to have mutable attributes during usage.

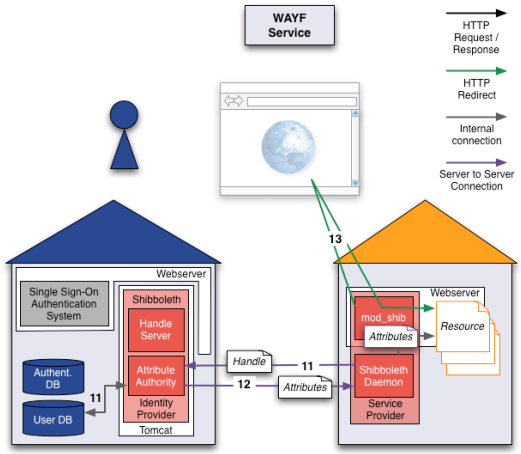

Figure 2 – Shibboleth System [17]

Shibboleth has two components to accomplish authentication, an Identity Provider and a Service Provider. The Identity Provider is local to the user's system and provides the user with a set of credentials as to their identity. The Service Provider is run by SPs to authenticate for their resources. The Service Provider has a list of Shibboleth-enabled resources, verifies the user's home system has rights to use that resource, and allows the user to use the system. The user's home system talks to the attribute database which has the responsibility to provide the attributes necessary to access the system.

The **GribShib** [20] project is collaboration between NCSA and the University of Chicago. It integrates the Shibboleth project with the Globus toolkit and the Globus Secure Infrastructure to create an authorization system that verifies the user and identifies a set of attributes through a centralized attribute server for that user (i.e. a pull-based system). The system then makes decisions based whether those attributes satisfy the local system requirements.

Once GridShib is installed as a plug-in for the Globus Toolkit, the toolkit has the ability to either use the pushed attribute assertions or to query a Shibboleth Identity Provider for the state of attributes. These grids are X.509-based and use Security Assertion Markup Language (SAML) to perform these pushes. This communication was in the first phase of the project, the second is to be the federation of name spaces and the management of trust configuration/metadata between the grid and the Shibboleth pieces [26].

The **VO Privilege Project** [7] was created by US CMS and US ATLAS "to develop and implement fine-grained authorization for access to grid-enabled resources and services in order to improve user account assignment and management at grid sites, and reduce the associated administrative overhead". This system is a role-based system and uses least privilege to assign attributes to a user. As opposed to GribShib, it is a push based system, contacting and receiving an attribute certificate to access resources within the grid.

The user specifies the role they want to run under when contacting the Virtual Organization Membership Service (VOMS). The VOMS issues a proxy certificate, which is forwarded to the gatekeeper of a resource. A gridmap callout interface invokes a module that extracts the appropriate attributes for verification with the system. The Site Authorization Service (SAZ) is then checked for site-specific rules about the attributes before the gatekeeper allows access to the resource. Like the implementation of the GribShib project, once the verification takes place, no further resource checks are done based on usage.

3. USAGE-BASED ACCESS CONTROL

Several models have been recently presented with respect to access control, most recently the RBAC [1][2][3] model and the $UCON_{ABC}$ [13] model. These models investigate the general case for access control, providing mechanisms to manage resources within a dynamic environment.

$UCON_{ABC}$ is a decision making model determined by combining authorizations, obligations, and conditions. A key component to the $UCON_{ABC}$ model is flexibility and the ability to dynamically make access control decision on changing criteria. This component makes it well suited to the dynamic nature of grid computing.

$UCON_{ABC}$ defines several terms as a part of its decision making model. These include:

- *Subject*s
- *Objects*
- *Rights*
- *Authorizations*
- *Obligations*
- *Conditions*

The concepts of mutability is key to the introduction of this decision making model – i.e. that the criteria being evaluated can be modified during the usage of the resource and that the changes to this criteria may modify the decision. The next section discusses the possible changes to the above terms necessary to implement a usage monitor.

Park et al. [13] present several model definitions for usage-based access control. These model definitions handle the cases necessary to define decision points to implement mutability. The model definitions are defined using the previously-mentioned terms – let's look at those terms:

Subjects are entities associated with attributes in which several opportunities to update are identified within the model definitions. Examples of subjects within the grid environment would be a VO, subsets of the VO, sites participating in the VO and users that are a part of the VO. In these cases, the attributes assigned with the subjects could have mutable values that affect the authorization, such as an attribute providing the right to advertise a service as long the VO is under its quality of service contract numbers for that site and user. Once the user goes over it QoS requirement, the authorization decision is changed.

Objects are similar to subjects that they have attributes associated with them. Examples of objects within the grid environment are compute cycles, data sets, free data space, backup devices, and network streams. As with subjects, attributes assigned to objects could have mutable values that affect authorization decisions. An example would be a dataset that has an attribute allowing it to be streamed across the network as long as the site's QoS contract is not being violated. Once the usage at the site increases past a threshold, the user is no longer allowed to perform that service.

Rights are privileges associated with subjects. The existence of a right is determined at the time of the access, the usage monitor must be able to determine whether that right exists or not. Examples of rights within the grid environment would be the right to read a dataset, to advertise a service, to open a network channel between processes, and to start a process.

Authorizations are a set of functional predicates that are evaluated for a usage decisions. Built into these predicates are update functions on attributes, a usage monitor would have to be able to perform those updates. An example of an authorization in the grid environment would be the decision to allow a user to start a process using a specific dataset that requires N number of computing units.

Obligations are functional predicates that evaluate requirements for usage decisions. These requirements may change based on usage, the usage monitor must be able to handle both changes to the requirements as well as changes to the satisfaction of the requirement during

usage. An example of an obligation within the grid environment would be the requirement to belong to both virtual organization A and virtual organization B to access dataset C. Obligations could have mutable values as well. An example of a mutable obligation would be losing the authorization to access one data set when there is an obligation to have access to both data sets to access either.

Like attributes, system *conditions* may change during usage, or requirements for system conditions may change. The usage monitor must handle those situations. An example of conditions within the grid environment would be accessing a dataset only within certain hours to not impact another subjects performance per a quality of service contract. Conditions have mutable values, an example would be changing the hours compute cycles are available based on immediate maintenance needs.

4. DIGITAL RIGHTS MANAGEMENT

The availability of online digital content has dramatically increased in recent years, leading to Digital Rights Management (DRM) technologies intended to protect the rights associated with that content. DRM is broadly defined by Sohn [21] as "technical measures used to protect content in digital media devices and services".

Sohn compares the tradeoffs between DRM systems by identifying four key factors for assessing and comparing DRM.

- Transparency – what do the users see? Are they affected by the technology?
- Effect on use – how does the access control technology affect the use of the content?
- Collateral impact – what other affects does the DRM impose on the users of the technology?
- Purpose and consumer benefit – Is DRM allowing new consumer technology to flourish? Or is it being used to lock consumers into current technology?

We next look at how these metrics apply within the grid environment.

Transparency. One of the key issues with grid middleware is the inflexibility of the administration and management. Current access control is done by multiple logins or by assigning all grid users to a common login. The combination of the authentication system – which provides the attributes for the user – and the usage monitor should work as transparently as possible. This transparency includes performance, providing on-going usage decisions must not be overly cumbersome to the user.

Effect on use. The system should not present side effects of its implementation that adversely affect the user. If all the usage based attributes, conditions, and obligations are met, the user should be able to access the objects without issue.

Collateral impact. The combination of the authentication system and the usage monitor should have little to no side affects on other processes running within the local environment. This would include local environments that belong to multiple domains or VOs and whose resources may be shared between those domains.

Purpose and consumer benefit. The authentication and usage monitor systems should be standalone as much as possible. The systems should avoid tightly coupled interfaces to allow flexibility in growth and expansion.

From this, we show DRM and grid computing access control share many of the same requirements for their implementations. Sohn has identified generic requirements for DRM, we next explore specific systems to determine similarities in requirements for implementation. Michiels et al. [27] identified several key implementations for DRM.

4.1 DRM Implementations

Windows Media DRM [22] is a license-based access control model, encrypting the content with a key and packaging that key, along with other pertinent metadata, into a license. The

license is acquired automatically by the player from a license server when the user purchases or rents the media. The player then uses the key to unencrypt the content and play it.

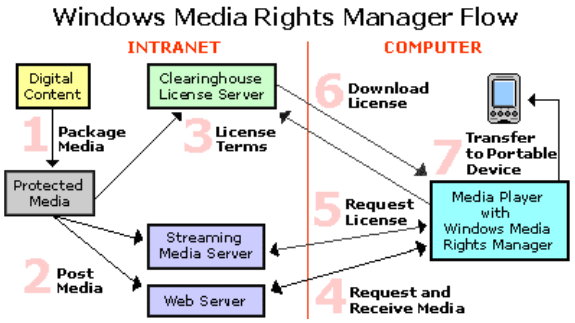

Figure 3. Windows Media DRM

The metadata the licenses contain includes the number of access times, characteristics about the access devices, start and end times for the licenses, the rights associated with the license, etc. Licenses can be delivered pre-packaged, after attempted access, or on request.

In this model, the player acts as the usage monitor, ensuring that attributes established by the license are enforced, that attributes like play count are decremented appropriately, and that obligations are met before usage.

Helix DRM [23] or Helix Security Manager provides access control for files to be presented in streaming format or for download. The architecture consists of Security Manager**,** a Java-based application that generates secure URLs, and the Security Adaptor, a server-side program performing validation of the URLs.

Helix is focused on streaming delivery, and uses the secure URL, which is appended on the original URL, for one-time authorization. All attribute management is handled by the server, as well as all usage monitoring.

Aegis DRM [25] addresses the protection of digital content usage within a company, government entity, or other organization. The software consists of a license server that handles licenses for aoftware distribution, an access control server which keeps a centralized list of users and access rights, and a module (called *Protector*) that allows content to be created and defines how the content will be allowed to be used.

All of the current DRM systems I examined used the license-based systems in which the user is issued a license in some form, which contains a key that unlocks the encrypted media. The role of the usage monitor is performed either in a centralized mechanism such as a license or access server, or by the player enforcing the usage rules encrypted in the metadata.

5. USAGE MONITOR REQUIREMENTS

From our analysis of the UCON model definitions, we determined the usage monitor:

Requirement 1. The usage monitor must be able to communicate with the attribute server seamlessly to update attributes and attribute characteristics.

A user has an attribute that he/she can only access a specific dataset N time. A common replication technique is to have replicas of a dataset throughout the grid; this is not as easy as decrementing the usage count on the specific system. Every time the user accesses the dataset, the monitor must update the attribute server that the user has done so using this replica.

Requirement 2. The usage monitor must be able to communicate with the requesting process in a way to deliver on-going usage decisions.

In the previous example, the access decision could be made at the time the request to access a dataset was made. Looking the situation where the decision was made on a mutable attribute, this becomes more complex. If a process can only access a dataset while the user is below their QoS requirements, the monitor must have some way to notify the process to suspend

operations until the QoS metric is below the required benchmark.

Requirement 3. The usage monitor must be able to determine rights based on the usage decision and to retain those rights for future decisions.

Authorization decisions are based on determining whether a subject has a set of rights to perform an operation on an object given the conditions and satisfaction of the obligations. Some of these – such as an obligation to belong to a complete set of VOs – only need be evaluated once, to continue to do so would cause performance issues for the monitor. Once the obligation has been evaluated once for this user, it would remember that the evaluation had occurred and not perform it again.

Requirement 4. The usage monitor must be able to identify and evaluate on-going obligations. Must be able identify and evaluate on-going conditions.

As a counterpoint to the above example, there are conditions and obligations that have to re-evaluated based on mutability. Using the QoS example again, if the QoS metrics established by the site are exceeded for one VO, the monitor must be able to notify the processes running for that VO to suspend operations until the lower watermark is reestablished and operations for that VO can continue.

From our examination of DRM models and existing technologies, the usage monitor:

Requirement 5. The usage monitor must be transparent, providing the ability to perform on-going access decisions without intruding into the users awareness.

When a user has selected to have a process run accessing a specific dataset when the QoS metrics are below a specific point, the monitor should start and suspend the process when the start and suspend characteristics are met without the user being required to have special code to handle that situation.

Requirement 6. The usage monitor must no present side effects of its implementation that adversely affect the user.

Using the previous example, a user has selected to run on the grid when a set QoS metrics are met, and to be suspended when these metrics are not satisfied. When the user is restarted, one of the datasets the user was previously using has gone off-line for maintenance. The monitor should keep the job suspended until the resources the user is using are restored to their previous state.

Requirement 7. The usage monitor must not have side effects on other processes or applications running within the local environment.

If a process is suspended for QoS constraints, the suspension and restarting of that process should not affect another processes authorization. In the previous example, a process is started and suspended based on QoS conditions. If a condition exists that only one process can access the dataset at a time, and that policy supersedes the QoS policy, the monitor should not allow the follow-on process to begin operations.

Requirement 8. The usage monitor must not lock the user into a specific technology.

If a user is using an authorization such as Shibboleth and the VO switches to a different authorization system – the user should not have to make changes to the application to get the same functionality from the monitor – assuming the monitor supports the new authorization system.

Finally, from our analysis of the existing grid projects, we identify the usage monitor:

Requirement 9. The usage monitor must be compatible with multiple attribute-based authentication systems

The usage monitor must be built in a way that allows multiple authorization systems to be implemented and co-exist. This keeps the

problem with the previous example from being exacerbated.

Requirement 10. The usage monitor must be able to run on multiple architectures and environments.

Grids are often made up of heterogeneous systems – systems of multiple architectures and multiple operating systems. If a user writes an application that runs using C and Linux on one system and C++ and Ubuntu on a different system, the monitor must be able to support it. This turns out to be more challenging that first blush as a key component of the monitor must be the interaction with the QoS metrics.

Requirement 11. The usage monitor must be able to interface with virtual environment implementations.

One of the most common ways of implementing grid computing environments these days is through the use of virtual environments. The usage monitor must not have a different operation or interface to the user if the environment of the specific grid is implemented through virtual environments. This would cause problems for a user that belongs to multiple VOs, each running different grid architectures. These effects should be minimized.

Requirement 12. The usage monitor must be able to handle multiple VOs accessing the same set of resources.

If a site belongs to VO A and to VO B, the conditions specified to the monitor should be done in a way that multiple accesses to the same resource or by a user in multiple domains does not conflict with each other. Like the requirement for multiple architectures, the monitor must be able to handle the QoS requirements for multiple organizations.

6. RELATED WORK

The primary security administration model of today is Role Based Access Control (RBAC). RBAC was initially presented in 1992 by Ferraiolo and Kuhn [1]. In 2000, the Ferraiolo / Kuhn model was merged with the framework defined by Sandhu et al [2] to create the NIST RBAC model [3]. This model was then standardized in 2004.

RBAC has become the predominant mechanism due to its ease of use and lowering of cost. Most commercial vendors use the model to implement their database and/or distributed network security. One of the key detractions of RBAC however, is the static nature of its access control. Once the decision-making point has passed, current circumstances are no longer taken into account. This lack of mutability has led to the introduction of models that resolve that issue.

Zhang et. al [10] have presented a usage-based attribute model based on the UCON and PEI models, where the UCON model works within the Policy framework of the PEI model. They present a design architecture to provide enforcement in the context of the Grid Security Infrastructure (GSI). Three architectural units are defined – user platforms, individual resource providers (RPs), and an attribute repository. They also define a usage monitor to run on each RP. Their prototype implements this using DB4Object as an object-oriented database to store these object attributes. While this handles a subset of the mutable attributes, a more complete implementation is offered by this paper.

DRM is an access control technology that takes usage into account. Examples of DRM systems examined were Windows Media DRM [22], Helix DRM [23], Lightweight DRM [24], and Aegis DRM [25]. Requirements gathered from these systems pertained mainly to usability of the usage monitor as they are very user-focused.

7. CONCLUSION AND FUTURE WORK

In this paper we presented the need for a usage monitor to enforce usage based attribute systems that are currently being proposed or are under development. A background of grid computing, including the current state of authorization was given, as well as an examination of two current grid authorization projects, GridShib and VO

Privilege. We identified the requirements for a usage monitor from the work by Zhang et al. on a usage-based control model. We also examined the current implementations of DRM models for requirements, such as HelixDRM, Windows Media DRM, and Aegis DRM.

After these examinations, a set of comprehensive requirements for the functionality of the usage monitor was summarized. In addition to the specific requirements identified in Section 5, the monitor will have the requirements of portability, extensibility, and maintainability. These requirements will be the building block for the next phase of the project – to build a framework for our access control usage monitor.

Enforcement of access control policies is always one of the most difficult areas of an implementation. These requirements will be be the building block for the next phase of the project – to build a framework for our access control usage monitor that provides the enforcement capability needed for this technology to be successful.

REFERENCES

[1] Ferraiolo, D., Kuhn, R. Role-Based Access Control. 1992. *In Proceedings of the 15th NIST/NCSC National Computer Security Conference.*

[2] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. Role-Based Access Control Models. 1996. *IEEE Computer, Volume 29, Number 2, pp. 38-47.*

[3] Sandhu, R., Ferraiolo, D., and Kuhn, R. 2000. The NIST model for role-based access control: Towards a unified standard. *In Proceedings of 5th ACM Workshop on Role-Based Access Control* .

[4] Casanova H. Distributed Computing Research Issues in Grid Computing. September 2002. *ACM SIGACT News, Vol. 33, Issue 3*, pp 50-70.

[5] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. 2001. *International Journal of High Performance Supercomputing Applications, 15(3).*

[6] D. Menasce, E. Casalicchio. Quality of Service Aspects and Metrics in Grid Computing. July/August 2004. *IEEE Internet Computing, Vol. 8, No. 4.*

[7] VO Privilege project home. http://www.fnal.gov/docs/products/voprivilege/

[8] Calder, B., Chien, A., Wang, J., Yang, D. The Entropia Virtual Machine for Desktop Grids. 2005. *In Proceedings of the First Virtual ACM/USENIX Conference on Virtual Execution Environments (VEE'05)*

[9] Alpern, B., Auerback, J., Bala, V., Frauenjofer, T., Mummert, T. Pigott, M. 2005. *In Proceedings of the First Virtual ACM/USENIX Conference on Virtual Execution Environments (VEE'05)*

[10] Zhang, X., Nakae, M., Covington, M., Sandhu, R. Toward a Usage-Based Security Framework for Collaborative Computing Systems. 2008. *ACM Transactions on Information and System Security. Vol. 11, No. 1, Article 3.*

[11] Open Sciences Grid. http://www.opensciencesgrid.org/

[12] http://www.globus.org/toolkit

[13] J. Park and R. Sandhu, *The UCON_ABC Usage Model*. 2004. ACM Transactions on Information and System Security, 7(1): 128-174.

[14] http://www-03.ibm.com/grid/pdf/innovperspective.pdf. p. 18.

[15] http://shibboleth.internet2.edu/.

[16] http://www.internet2.edu/resources/AboutInternet2.pdf.

[17] http://www.switch.ch/aai/demo/expert.html.

[18] http://www.teragrid.org/.

[19] http://www.earthsystemgrid.org/.

[20] http://gridshib.globus.org/.

[21] Sohn, D. Understanding DRM. November/December 2005. *ACM QUEU*. pp. 32-39.

[22] Microsoft Windows Media Rights Manager. http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx.

[23] Real Networks. Helix DRM. http://www.realnetworks.com/products/drm.

[24] Lightweight DRM. http://www.lightweightdrm.org/.

[25] Aegis DRM. http://www.aegisdrm.com/.

[26] Gridshib. http://spaces.internet2.edu/display/GS/ProjectSummary.

[27] Michiels, S., Verslype, K., Joosen, W., Decker, B. Towards a Software Architecture for DRM. 2005. In *Proceedings of the Workshop for Digital Rights Management (DRM '05).* Pp. 65-74.

[28] Wikipedia entry for Virtual organization. http://en.wikipedia.org/wiki/Virtual_organization.

[29] Wikedia entry for DRM. http://en.wikipedia.org/wiki/Digital_rights_management.