DYNAMIC CONTEXT AWARE ACCESS CONTROL FOR GRID APPLICATIONS

BY GUANGSEN ZHANG

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Manish Parashar

and approved by

_____

_____

_____

New Brunswick, New Jersey

December, 2003

ABSTRACT OF THE THESIS

Dynamic Context Aware Access Control for Grid Applications

by Guangsen Zhang

Thesis Director: Professor Manish Parashar

While the primary objective of Grid Computing is to facilitate the sharing of resource and service spanning across largely distributed and heterogeneous system, the success deployment of Grid infrastructure will make lots of applications possible. The applications range from pure scientific computing to commercial utilization. It will enhance the human creativity by increasing the computing capability and performance; allow geographically distributed people and computers to collaborate. The Grid infrastructure presents many challenges due to its inherent heterogeneity, multi-domain characteristic, and highly dynamic nature. One critical challenge is providing authentication, authorization and access control guarantees. Although lots of researches have been done on different aspects of security issues for Grid computing, these efforts focus on relatively static scenarios where access depends on identity of the subject. They do not address access control issues for pervasive Grid applications where the access privileges of a subject not only depend on its identity but also on its current context (i.e. current time, location, system resources, network state, etc.) and state. In this thesis, we present the SESAME dynamic context-aware access

control mechanism for pervasive Grid applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underlying dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC). We also present a prototype implementation of SESAME and DRBAC with the Discover computational collaboratory and an experimental evaluation of its overheads.

Keywords: Grid security, authorization and access control, context-aware, pervasive applications, Grid computing.

Acknowledgements

First and foremost, I thank Professor Manish Parashar, my advisor, for his guidance, patience, and encouragement during this research. I owe much gratitude to my parents for their unconditional support and love. I am especially grateful to my dear wife, Xiaokun Wang, for her love at all times. She is always emotionally supportive, and she helped improve the presentation of this thesis.

I also acknowledge the help from the members of TASSL Laboratory. Specifically, Viraj Bhat and Vincent Matossian.

Finally my special thanks to CAIP and all its staff, who have always promptly helped me in resolving both administrative and systems problems.

Dedication

To my Parents and my Wife

Table of Contents

Abstract ii

Acknowledgements i

Dedication ii

List of Tables v

List of Figures vi

1. Introduction 1

1.1. Motivation 1

1.2. Problem Statement 1

1.3. Contribution 3

1.4. Organization of the Thesis 4

2. Background 5

2.1. Security Service For Grid Applications 5

2.2. Security Requirement for Pervasive Grid Applications 9

2.3. Scenarios Need Dynamic Access Control 10

2.3.1. User to User Collaboration. 10

2.3.2. Component to Component Interaction 11

2.3.3. User Collaborate Access Resource 12

2.4. Access Control Model 13

2.4.1. Access Control Policies 13

2.4.2. Access Control Mechanisms 15

3. Dynamic Role based Access Control 17
3.1. RBAC 17
3.2. Dynamic Role based Access Control Model 19
3.3. DRBAC Operation 22

4. SESAME/DRBAC Prototype Implementation 24
4.1. Prototype Implementation 24
4.2. Prototype Operation in Discover 28

5. Experimental Evaluation 32

6. Summary and Conclusions 35
6.1. Discussion 35
6.2. Conclusion 36
6.3. Future Work 36

Appendix A. Role Based Access Control Models 38
References 40

List of Tables

4.1. Permission assignments for the example. 28
4.2. Permission definition for the example. 28
5.1. Interaction time in ms. for different context event frequencies. 33
5.2. Interaction time in ms. for different number of roles. 33
5.3. Interaction time in ms. for different number of permissions. 33

List of Figures

3.1. The dynamic access control model 20
3.2. Role hierarchy state machine 22
3.3. Permission hierarchy state machine 23
4.1. Discover architecture 24
4.2. Dynamic access control in discover 26
4.3. Sample RoleTransition policy in XML 27
4.4. Role and permission hierarchies for the example. 29
4.5. Permission hierarchy for the application 30
4.6. Dynamic access control in discover 31
A.1. Role Based Access Control Model: RBAC0 38
A.2. Role Based Access Control Model: RBAC1 39
A.3. Role Based Access Control Model: RBAC2 39

Chapter 1
Introduction

1.1 Motivation

Grid computing is rapidly emerging as the dominant paradigm of wide area distributed computing [8]. It's primary objective is to provide a service oriented infrastructure that leverages standardized protocols and services to enable pervasive access to, and coordinated sharing of geographically distributed hardware, software, and information resources. The Grid community and the Global Grid Forum [7] are investing considerable effort in developing and deploying standard architectures and protocols that enable seamless and secure discovery, access to, and interactions among resources, services, and applications. This potential for seamless aggregation, integration, and interactions has made it possible to conceive a new generation of Grid applications that are based on ad hoc, symbiotic and opportunistic interactions, where users, application components, Grid services, resources (systems, CPUs, instruments, storage) and data (archives, sensors) interact as peers. However, realizing such a pervasive Grid infrastructure presents many challenges due to its inherent heterogeneity, multi domain characteristic, and highly dynamic nature. One critical challenge is providing authentication, authorization and access control guarantees.

1.2 Problem Statement

The Grid Security Infrastructure(GSI) [9] has been accepted as the primary authentication mechanism for the Grid. Developed as part of the Globus project [22], GSI

defines single sign on algorithms and protocols, cross domain authentication protocols, and temporary credentials called proxy credentials. GSI is widely used and has been integrated into a number of Grid environments and applications. However, the authorization and access control challenges are not fully addressed by existing approaches. The Akenti [28] access control system enables multiple owners and administrators to define fine grained usage policies in a widely distributed system. The Akenti policy engine then gathers use conditions certificate defined by the resource owners and attribute certificates from the various stake holders, and grants access to a resource by matching of these two certificates. In the Community Authorization Service (CAS) [13], resource providers grant access to a community accounts as a whole. The CAS server is designed to maintain authorization information for all entities in the community. It keeps track of fine grained access control information and grants restricted GSI proxy certificates (PCs) to community members. M. Lorch et al [19] propose a fine grained authorization services to support ad hoc collaborations using attribute certificates. Similarly, L. Ramakrishnan et al [14] present an authorization infrastructure for component based Grid applications by providing authorization at the component interface.

While these research efforts listed above do address important aspects of the overall authorization and access control problem in a Grid environment, these efforts focus on relatively static scenarios where access depends on identity of the subject. They do not address access control issues for pervasive Grid applications where the access capabilities and privileges of a subject not only depend on its identity but also on its current context (i.e. current time, location, system resources, network state, etc.) and state. For example, consider a user accessing a remote resource or a data archive using a pervasive portal on her PDA. In such an application, the user's access privileges depend on who she is, where she is (in a secure or insecure environment), her context (current connectivity, current load), the state of the resource or data archive she is accessing, etc. Furthermore, her privileges will change as her context

changes, for example, if she moves from a secure wireless link to an insecure one. Similarly, when a Grid service interacts with another service on the Grid, the access privileges of the service will also depend on the credential of the service as well as the context and state of the service, which are dynamic.

1.3 Contribution

In this thesis, we present the SESAME [1] dynamic context aware access control mechanism for pervasive Grid applications [29, 30]. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underlying dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC) [25, 5], while retaining its advantages (i.e. ability to define and manage complex security policies). The model dynamically adjusts *Role Assignments* and *Permission Assignments* based on context information. In DRBAC, each subject is assigned a role subset from the entire role set by the authority service. Similarly, each object has permission subsets for each role that will access it. During a secure interaction, state machines are maintained by delegated access control agents at the subject (*Role State Machine*) to navigate the role subset, and the object (*Permission State Machine*) to navigate the permission subset for each active role. These state machines navigate the role/permission subsets to react to changes in context and define the currently active role at the subject and its assigned permissions at the object.

A prototype of SESAME and the DRBAC model has been implemented as part of the Discover [27, 20] computational collaboratory. Discover enables geographically distributed scientists and engineers to collaboratively access, monitor and control applications, services, resources and data on the Grid using pervasive portals. The feasibility, performance and overheads of SESAME are experimentally evaluated.

[1] Scalable, Environment Sensitive Access Management Engine

1.4 Organization of the Thesis

The thesis is organized as follows. The first part is introductory in nature. Chapter 2 provides a comprehensive description of concepts and notions in access control. It provides background information for the discussion presented later in the thesis. Chapter 3 presents the SESAME dynamic access control model and describes its operation. Chapter 4 describes the prototype implementation within the Discover collaboratory. Chapter 5 presents an experimental evaluation. I conclude with a discussion of contribution of this thesis and possible future direction in Chapter 6.

Chapter 2
Background

This Chapter presents the concepts underlying this thesis and the necessary materials for understanding the rest of the thesis. The presentation here consists of four parts. The first part introduces the related work in the Grid applications security, with a focus on authorization and access control aspect. The second part discusses the requirement of the authorization and access control for pervasive Grid applications. In the third part, the scenarios that need dynamic access control in pervasive Grid applications are discussed. Finally, the access control concepts and models are discussed.

2.1 Security Service For Grid Applications

GSI [23] is an alternative approach to inter domain security. It was developed under the Globus research project to support distributed computing environments, or Computational Grids, which are similar to virtual organizations. GSI deals with inter domain operations, bridging the different local security solutions of constituent sites. GSI includes several significant features:

- Credentials, using standard X.509v3 certificates as the private keys, represent the identity of each entityuser, resource, programspecifying the entitys name and additional information, such as a public key. A certification authority (CA), a trusted third party, ties an identity to a public private key pair by signing a certificate.

- An authentication algorithm, defined by the Secure Socket Layer Version 3 (SSLv3) protocol, checks the entitys identity. The veracity of an entitys identity is only as good as the trust placed in the CA that issued the certificate, so the local administrator installs these certificates, which are then used to verify the certificate chains.

- An entity can delegate a subset of its rights, such as a process a program creates to a third party by creating a temporary identity called a proxy. Proxy certificates can form a chain, beginning with the CA and growing, as first the user, then the users proxies, sign certificates. By checking the certificate chain, processes started on separate sites by the same user can authenticate to one another by tracing back along the certificate chain to find the original user certificate.

- Each resource can specify its policy for determining whether to accept incoming requests. The initial GSI used a simple access control list, but the current version uses other techniques.

- The authentication protocol verifies the global identity of involved parties, but GSI must convert this name to a local subject name such as a login name or Kerberos principal before the local security system can use the name. GSI does this by consulting a simple text based map file under the local sites control that defines the binding between global and local names.

- The standard interface GSS API provides access to security operations. GSI uses OpenSSL or SSLeay, the free implementation of SSLv3, for its authentication protocols and support for proxy certificates. SSLv3 is used widely for Web security, has been well scrutinized for security problems, and has broad acceptance as a mature protocol.

Akenti [**28**] is a distributed access control system designed by Ernest Orlando Lawrence Berkeley National Laboratory. The immediate motivation is to enable sharing over open networks of valued resources within the scientific community. The resources consist of large scientific instruments such as electron microscopes or high energy light sources; supercomputers or other high end computer servers etc. In such an environment, the stake holders of all kinds of resources have flexibly specify access requirements for their resource, and each resource may have multiple stake holders who will impose different conditions for access. Further, the principals in these scientific collaborations are geographically distributed and multi organizational. To solve these issues, Akenti use a policy based access control based on certificates, that convey identify, authorization, and attributes. Users are authenticated by presenting an X.509 identity certificate and proving that they know the associated private key. These certificates are issued by certificate authorities(CA) that verify the connection between a person or system component and possession of a public key/private key pair . Stake holders create and digitally sign use condition certificates that define conditions that must be satisfied by a user before being given access to resource. User attributes are asserted by "authorities" that provide assured information as digitally signed attribute certificates. Both use condition and attribute certificates may be stored local to the issuer as long as they can be provided by a server when they are needed to determine permissions during an access request. The heart of Akenti system is the Akenti policy engine, which gathers and verifies certificates and then evaluates the user's right to access the requested resource based on these certificates. As the certificates are distributed over the network, to search these certificates, a minimal authority file is stored with the resources need access control. This file contains a list of servers which supply identity and attribute certificates; the list of the use condition issuers; and where the use condition certificates are stored. In addition, there is a root authority file that contains the list of trusted CAs, and their public keys, for the whole resource tree. The Akenti policy engine searches each of the certificate directories

listed in the authority file. The user's access on resource based on the evaluation of these certificates. To improve the performance of searching the certificates across the network, Akenti take some ways: Filtering at server side to reduce the amount of certificates returned to Akenti, using hash search; caching certificates locally once they have been found and verified. Akenti has some vulnerabilities: if the certificate needed for some reason unavailable , the access control decision will be wrong. So the reliability of Akenti system will based on the reliability of using certificates across the distributed system.

CAS [13] is a community Authorization Service for group collaboration. It focus on supporting the centralized specification of community policies governing collections of resources. A community runs a CAS server to keep track of its membership and fine grained access control policies. A user wishing to access community resources contacts the CAS server, which delegate rights to the user based on the request and the user's role within the community. These rights are in the form of capabilities, which users can present at a resource to gain access on behalf of the community. The CAS architecture build on public key authentication and delegation mechanisms provided by Grid Security Infrastructure(GSI). In GSI, an entity can delegate a subset of its rights to a third party by creating a temporary identity called a proxy. In order to support CAS, the ability of proxy is extended by carry policy information restricting its use, such a proxy is called a restricted proxy. This structure address the scalability problem by reducing the necessary trust relationships from U*R(U: user, R: resource) to U+R: each user needs to be known and trusted by the CAS server, but not by each resource; each resource need to be known and trusted by the CAS server, but not by each user. The CAS server provides a centralized location at which various use conditions that govern access to a resource can be collected; once these are verified, the resource need deal only with a single capability. Also CAS provides a mechanism to delegate permissions on a set of resources distributed across different administrative domains. The vulnerability of CAS is if a CAS server is compromised, it can issues

credentials that don't reflect the policies of the community that it represents. Because CAS is a centralized mechanism, the CAS server will be the bottleneck of the whole system.

2.2 Security Requirement for Pervasive Grid Applications

Pervasive Grid applications require all of the standard security functions, including authentication, access control, integrity, privacy, and non-repudiation. Among them, authentication, authorization and access control are critical to the Grid computing infrastructure. While The Grid Security Infrastructure has been accepted as the primary authentication mechanism for the Grid, the authorization and access control challenges are not fully addressed by existing approaches. The Akenti access control system enables multiple owners and administrators to define fine grained access control policies in a widely distributed system. The Akenti policy engine then gathers use conditions certificate defined by the resource owners and attribute certificates from the various stake holders, and grants access to a resource by matching of these two certificates. In the Community Authorization Service (CAS), resource providers grant access to a community accounts as a whole. The CAS server is designed to maintain authorization information for all entities in the community. It keeps track of fine grained access control information and grants restricted GSI proxy certificates (PCs) to community members. Other works include the authorization service to support ad hoc collaborations using attribute certificates the authorization infrastructure for component based Grid applications by providing authorization at the component interface. While these research efforts address important aspects of the overall authorization and access control problem in a Grid environment, these efforts focus on relatively static scenarios where access depends on identity of the subject. They do not address access control issues for pervasive Grid applications where the access privileges of a subject not only depend on its identity but also on its current context

(i.e. current time, location, system resources, network state, etc.) and state. Our approach is SESAME (scalable environment sensitive access management engine). It complements current authorization mechanisms to dynamically grant permissions to users based on their current context. The underlying dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC), while retaining its advantages (i.e. ability to define and manage complex security policies).

2.3 Scenarios Need Dynamic Access Control

Grid applications frequently involve many more entities, dynamic collaboration, dynamic system environment. In this section, we summarizer three categories of scenarios in Grid computing that need dynamic access control mechanism: User to User collaboration, Component to Component Interaction, Users collaborate access resource.

2.3.1 User to User Collaboration.

In this scenario, users from different domain will collaborate together to carry out a work. For example, thousand of physicists at hundreds of laboratories and universities worldwide come together to design, create, operate, and analyze the products of a major detector at CERN, the European high energy physics laboratory. During the analysis, the scientists will access the computing result and data storage data of other scientists [8]. Each scientist has his or her own access control policies to protect the security of the resource; however, the access control should be dynamically changed to meet the dynamic changing environment.

- The pervasive access property of the Grid computing make the scientist collaborates with other scientists from anywhere possible. With the mobile device, the scientist can move to location that is not secure enough, the peer scientist who provide resource access maybe want deny the further access in this situation.

- The network bandwidth is limit, in some situation, if too many scientists access the resource of one scientist simultaneously, the network will be jammed. Access control should provide access only to critical users at this situation and deny the access of others. But the access control mechanism should also be capable of granting the access to those noncritical users while network is in light load.
- During the collaboration, some scientists will dynamically join the collaboration; the access control mechanism of each scientist should adjust dynamically according to the local policy to protect the confidential of the data.
- When scientist A collaborate with other remote peer scientists, the remote peer want know whether scientist A are in secure environment, for example, any unknown person around A. And the permission will be granted to A dynamically according to A's environment.

In these scenarios, the identity of the user is not the only credential to make access control decision. We need other information such as location, network usage, people's social environment to grant permission. As this information is dynamic and will change from time to time, observably, dynamic access control mechanism is necessary.

2.3.2 Component to Component Interaction

By dynamically composting the autonomic components, we can build autonomic applications upon Grid computing infrastructure. The autonomic components have the properties of self configuring, self healing, self optimizing and self protecting [1, 15]. They can perceive the changes in the environment and adjust their own behavior to adapt to the new environment. The autonomic applications will composite these autonomic components dynamically according to the rules predefined(highest performance, lowest cost, reservation, execution time upper bound, best accuracy). As autonomic component is context aware, they can know the environment around them.

They need change their access control policy as illustrated by the scenarios listed below:

- The component A wants access the resource of component B while component C is accessing the resource of B. As A has higher priority to use the resource of B, B need change the privilege it grants to C.
- Component A is involved in an application. During the process of work flow, new Components are included in the application. As A has no trust relationship with the new component, it will restrict other components to access the critical service in case of eavesdropping of new component.
- While Component A is providing service to other components, its local system resource will be used up. A should deny some components' access privilege to keep the service quality.

In autonomic computing, all the components will be context ware, so component can use the context information to provide dynamic access control

2.3.3 User Collaborate Access Resource

The Grid computing infrastructure will aggregate computational and information resources from different organizations in widespread locations. Users can access the resource from anywhere. The different organizations have different local security policy and the network links have different security level at different location. The Access control mechanism in such system will be more complex than the traditional distributed system [11]. The network topology will be dynamically updated, the new resource will be dynamically included, and the user who accesses the resource can be anyone. In such a dynamic and heterogeneous environment, static access control mechanism based on the assumption that all the resource and user are known to the

system in advance will not be feasible. Dynamic access control mechanism is critical to the scenarios described below:

- The user runs an application at his office. Then the application will access the resource somewhere else. With the capability of single sign on, the application can use the users credential to access the resource. During this procedure, if the user moves to another site which is not secure, the delegate application will possible be denied to continue access some resource because the potential risk of leaking information to malicious user.

- The user runs certain simulation program, which will dynamically get service from resource distributed in the Grid. The resource will change the privilege granted to the user according to other resources that the user is accessing.

2.4 Access Control Model

The basic way to model access control is a four tuple: (S;O;A;M), where S is the set of subjects, O is the set of objects, A is the set of actions (access rights), and M is a function that maps a $tuple(s; o; a) \in S \times O \times A$ to a decision $\in \{T; F\}$. The mapping M can be stored in an access matrix, with rows corresponding to subjects, columns corresponding to objects, and matrix entries indicating allowed access rights. In practice, a typical access matrix is large and sparse, and it is difficult to store, manage, and understand such a matrix directly. Therefore, various access control policies have been developed.

2.4.1 Access Control Policies

Discretionary Access Control(DAC). DAC is A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission

is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control). DAC permits the granting and revoking of access privileges to be left to the discretion of individual users. This is based on the notion that individual users are "owners" of objects. Ownership is usually acquired as a consequence of creating the object.

Mandatory Access Control (MAC). MAC is defined as a means of restricting access to objects based on sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity. The different security levels in a system form a lattice. MAC is typically used to enforce one directional information flow in such a lattice. The rule for read access requires that a user with a given clearance level can only read information with the same or lower classification level. The rule for write access requires that a user with a given clearance level can only write information to a target with the same or higher classification level. This prevents a user from declassifying information without authorization.

Role based Access Control (RBAC). Recently, role based access control (RBAC) has emerged as a promising alternative to the two traditional classes of access control policies. The notion of role is central to RBAC. Permissions are associated with roles, and users are granted membership in appropriate roles, thereby acquiring the roles' permissions. More advanced RBAC models include role hierarchy and constraints. Roles are created for the various functions in an organization and are assigned to and revoked from users based on users' responsibilities and qualifications. The power of RBAC comes from the fact that the notion of "role" captures the way most organizations operate. RBAC by itself is policy neutral. It can be used to implement MAC. The notion of roles makes the creation and modification of security policies easier so that security policies can more easily evolve incrementally over the system life cycle. Roles add another layer between principals and objects, thus helping to model the many to many relationships between principals and objects. In

RBAC, authorization is decided on the basis of which role the principal is associated with and which access rights the role has. Thus the full identity information of a requester may not be needed in access control. As long as the requester can provide proof that it is associated with the correct role, the request can be allowed. One of the most important features of roles is that it enables us to reason about the role-role relationship that is traditionally difficult to do:

- Mutual exclusion. This is a static form of separation of duty. It limits the same user to be assigned to at most one role in a mutually exclusive set.
- Pre requisition. If a role is declared a prerequisite for another role in the system, it means that a user may belong to the latter role only if he/she already belongs to the first role.
- Role hierarchy. One role may inherit permissions from other roles. This forms a role hierarchy.
- Cardinality constraints. This imposes a limit on the number of users that may be assigned to the role.
- Various forms of Dynamic Separation of Duty. It refers to those role constraints to be enforced at run time .

2.4.2 Access Control Mechanisms

Techniques to implement the above access control policies include the following:

Access Control Lists (ACLs): An access control list is an attribute of a target object, stating which users can invoke which actions on it. An access control list specifies the contents of the column related to the target object in the access control matrix.

Capabilities: Capabilities are results of decomposing the access matrix by rows. In this scheme, associated with each subject is a list that defines the objects to which

the subject has access rights and what are those authorized rights. A capability is effectively a ticket, possessed by a requester, that authorizes the holder to access a specified object in specified ways. Some capabilities can only be used by a specified principal, while others may be transferred to other principals.

Security Labels: A security label is a set of security attribute information bound to a user, a target, or a piece of information in transmission. The label indicates the sensitivity level of the data. This mechanism is used to implement MAC.

Chapter 3
Dynamic Role based Access Control

As mentioned in previous chapter, a key requirement for pervasive Grid applications is the support for dynamic, seamless and secure interactions between the participating entities, i.e. components, services, applications, data, instruments, resources and users. Guaranteeing interaction security requires a fine grained access control mechanism. Furthermore, in the highly dynamic and heterogeneous Grid environment, the access privileges of an entity depend on its credential, context and current state, which are dynamic. In this Chapter, we present the SESAME Dynamic Role Based Access Control model(DRBAC) to address these requirements. The traditional Role Base Access Control(RBAC) model is first discussed. The DRBAC model and its operation are then described in detail.

3.1 RBAC

Role based access control (RBAC) [24, 10] is an alternative to traditional discretionary (DAC) and mandatory access control (MAC). In RBAC, users are assigned roles and roles are assigned permissions. A principle motivation behind RBAC is the ability to specify and enforce enterprize specific security policies in a way that maps naturally to an organization's structure. As user/role associations change more frequently then role/permission associations, in most organizations, RBAC results in reduced administrative costs as compared to associating users directly with permissions. It can be shown that the cost of administrating RBAC is proportional to U+P while the cost of associating users directly with permissions is proportional to U*P, where

U is the number of individuals in a role and P is the number of permissions required by the role [5]. Sandhu [25] defines a comprehensive framework for RBAC models which are characterized as follows:

- $RBAC_0$: the basic model where users are associated with roles and roles are associated with permissions. As shown in A.1.
- $RBAC_1$: $RBAC_0$ with role hierarchies. As shown in A.2.
- $RBAC_2$: $RBAC_1$ with constraints on user/role, role/role, and/or role/permission associations. As shown in A.3.

Recently RBAC has been found to be the most attractive solution for providing security in a distributed computing infrastructure [5]. Although the RBAC models vary from very simple to pretty complex, they all share the same basic structure of subject, role and privilege. Other factors, such as relationship, time and location, which may be part of an access decision, are not considered in these models. The SESAME DRBAC model presented in this thesis extends RBAC to provide context aware access control mechanisms for dynamic and pervasive Grid applications.

We also noted others who provide context aware access control mechanism by extending RBAC model. Michael J. Covington [17, 16, 18] has proposed the Generalized Role Based Access Control (GRBAC) model. In this model, he extends the traditional RBAC by applying the roles to all the entities in a system. (In RBAC, the role concept is only used for subjects). By defining three kinds of roles in the model: Subject roles, Environment roles, and Object roles, context information can be used as a factor in making access decisions. In GRBAC, the definition of environment roles allows the model to partially address problem we described, but it may not be feasible in practice because the potential large amount of environment roles make the system hard to maintain. Also, by defining too many roles in the system, it looses the advantage that the RBAC provides. Further more, in certain situations it may

fail. For example, in the Aware Home, if two users both have a parent role and one of them presented in the kitchen room, the environment role kitchen room will be activated for both of them thus granting wrong permissions to another user. Ultimately, the access request is either granted or denied based on the present of an environment role. This is not the case in pervasive Grid computing, where we always want the privilege to access the resource change continuously as the environment change. Our approach attempts to address this issues and is described in the following sections.

3.2 Dynamic Role based Access Control Model

The formalization of the DRBAC model is based on the RBAC model presented in [6]. The DRBAC model is illustrated in Figure 3.1. It has the following components:

- USERS. A user is an entity whose access is being controlled. USERS represents a set of users.
- ROLES. A role is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role. ROLES represents a set of roles.
- PERMS. A permission is an approval to access one or more DRBAC protected resources. PERMS represents a set of permissions.
- ENVS. ENVS represents the set of context information for the system. We use an authorized "context agent" to collect context information in our system.
- SESSIONS. A session is a set of interactions between subjects and objects. SESSIONS represents a set of sessions.
- UA. UA is the mapping that assigns a role to a user. In a session, each user is assigned a set of roles and the context information is used to determine the active role among these. The user accesses the resource using this active role.

- PA. PA is the mapping that assign permissions to a role. Every role which has privileges to access the resource is assigned a set of permissions and the context information is used to determine the active permissions for the roles.

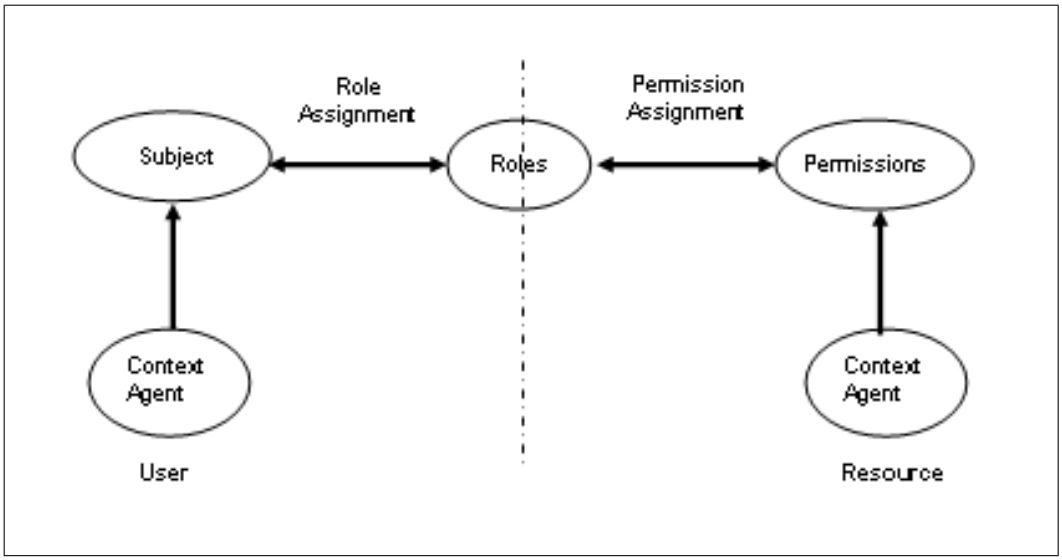

Figure 3.1: The dynamic access control model

In the DRBAC model, a Central Authority (CA) maintains the overall role hierarchy for each domain. When the subject logs into the system, based on her credential and capability, a subset of the role hierarchy is assigned to her for the session. The CA then sets up and delegates (using GSI) a local context agent for the subject. This agent monitors the context for the subject (using services provided by the Grid middleware) and dynamically adapts the active role. Similarly every subject maintains a set of permission hierarchies for each potential role that will access the resource. A delegated local context agent at the subject resource will use environment and state information to dynamically adjust the permissions for each role. We formally define the DRBAC model as follows:

- *USERS, ROLES, PERMS, ENVS and SESSIONS (users, roles, permissions, environments and sessions, respectively).*

- *ACT_ROLE and ACT_PERMISSION (active role and active permission respectively).*

- *UA⊆USERS×ROLES, a many-to-many mapping user-to-role assignment relation.*

- *PA⊆PERMS×ROLES , a many-to-many mapping permission-to-role assignment relation.*

- *Assigned_roles(u:USERS, e:ENVS)* $\rightarrow 2^{ROLES}$*, the mapping of user u onto a set of roles.*

- *Assigned_permissions(r:ROLES, e:ENVS)*$\rightarrow 2^{PERMS}$*, the mapping of role r onto a set of permissions.*

- *User_sessions(u:USERS)* $\rightarrow 2^{SESSIONS}$*, the mapping of user u onto a set of sessions.*

- *Session_roles(s:SESSIONS)* $\rightarrow 2^{ROLESS}$*, the mapping of session s onto a set of roles. Formally: session_roles(*s_i*)* $\subseteq \{r \in ROLES | (session\_roles(s_i), r) \in UA\}$

- *RH* $\subseteq$ *ROLES×ROLES is a partial order on ROLES called the inheritance relation, written as* $\geq$ *, where* $r_1 \geq r_2$ *only if all permissions of* r_2 *are also permissions of* r_1*, and all users of* r_1 *are also users of* r_2*.*

- *PH* $\subseteq$ *PERMS× PERMS is a partial order on PERMS called the inheritance relation, written as* $\geq$ *, where* $p_1 \geq p_2$ *only if all roles of* p_1 *are also roles of* p_2*.*

In the formal definitions above, UA (user assignment) defines the relationship among roles, users and environments; PA (permission assignment) defines the relationship among permissions, roles and environments. RH (role hierarchy) and PH (permission hierarchy) define the inheritance relationship among roles and permissions respectively. The following section explains the operation of our model in detail.

3.3 DRBAC Operation

In the DRBAC model, we assign each user a role subset from the entire role set. Similarly each resource will assign a permission subset from the entire permission set to each role that has privileges to access the resource. Figure 3.2 shows the relationship between the role hierarchy maintained at the Central Authority (CA) and the subset of this hierarchy assigned to a particular user.

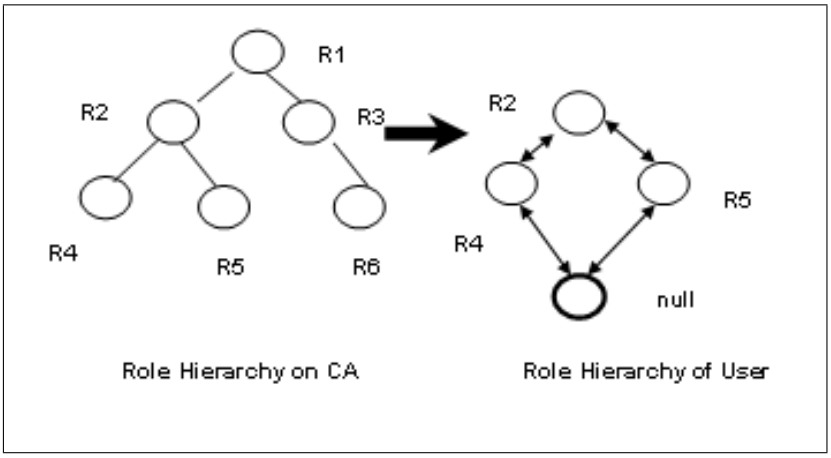

Figure 3.2: Role hierarchy state machine

We use state machines at the subject (Role State Machine) to maintain the role subset for a user, and at the object (Permission State Machine) to maintain the permission subset for each role. A state machine consists of state variables (a role or permission) that encode state, and events that transform its state. The delegated local context agent uses middleware services to monitor context and generates events to trigger a transition of the state machine when necessary.

A permission hierarchy is shown in the Figure 3.3. Note that the null permission signifies no access privileges. A transition is defined as *T(Initial State, Destination State)*. So *T(P1, P2)* represents the transition from P1 to P2 and *T(P2, P1)* represents the transition from P2 to P1. In this example, P2 is the current active permission. Role transitions in the Role State Machine are similarly defined.

Key concerns in the implementation of the proposed state machine based access

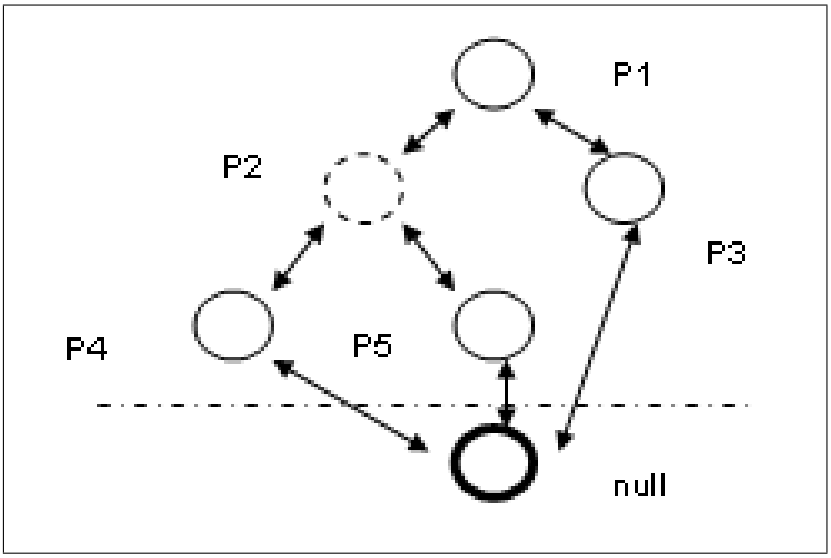

Figure 3.3: Permission hierarchy state machine

control mechanism include its performance overheads and the reliability and security of the context information. In a typical organization, the number of roles and permissions is relatively small, no more than 20. As a result, with the increasing computational capability of systems, maintaining the state machine will have little if any impact on performance. Also, there are a number of research and commercial efforts [2] developing context toolkits that can provide reliable and secure context services.

Chapter 4
SESAME/DRBAC Prototype Implementation

4.1 Prototype Implementation

A prototype of SESAME and the DRBAC model has been implemented as part of the Discover [27, 4] computational collaboratory. Discover is a Grid based computational collaboratory that enables geographically distributed scientists and engineers to collaboratively access, monitor, and control distributed applications, services, resources and data on the Grid using pervasive portal. The architecture of Discover is presented in Figure 4.1. Key components of the Discover collaboratory include:

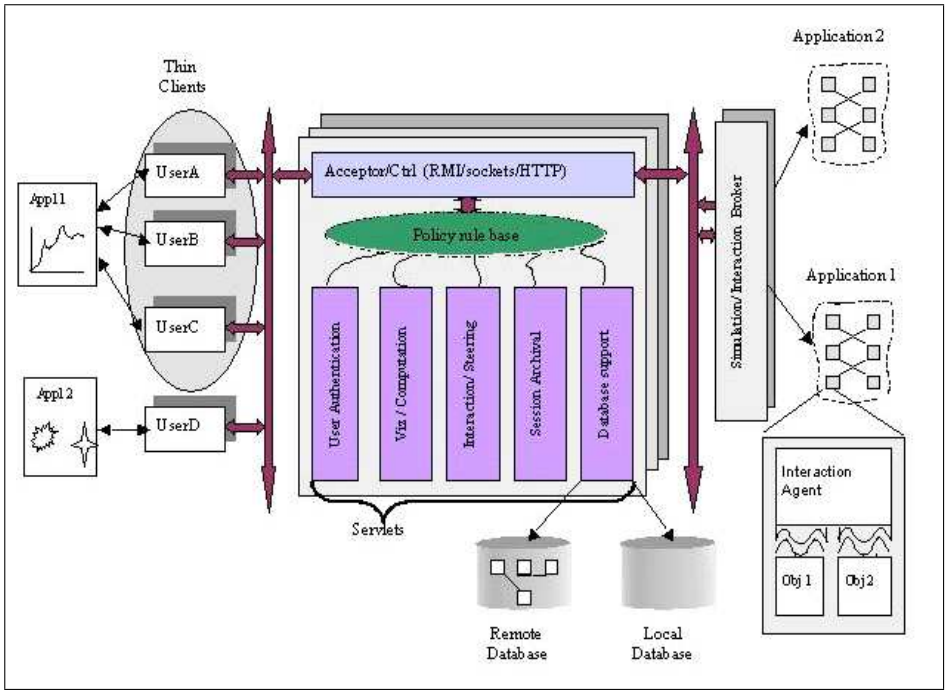

Figure 4.1: Discover architecture

- **Discover Collaborative Portals** [27] that provide users with pervasive and collaborative access to Grid applications, services and resources. Using these portals, users can discover and allocate resources, configure and launch applications and services, and monitor, interact with, and steer their execution.

- **Discover Middleware Substrate** [20, 4] that enables global collaborative access to multiple, geographically distributed instances of the Discover computational collaboratory, and provides interoperability between Discover and external Grid services such as those provided by Globus [22].

- **DIOS Interactive Object Framework (DIOS)** [21] that enables the runtime monitoring, interaction and computational steering of Grid applications and services. DIOS enables application objects to be enhanced with sensors and actuators so that they can be interrogated and controlled.

An overview of the integration of SESAME and DRBAC with Discover is presented in Figure 4.2. SESAME ensures the users can access, monitor and steer Grid resources/applications/services only if they have appropriate privileges and capabilities. As Discover portals are pervasive and the Grid environment is dynamic, this requires dynamic context aware access management. Note that authentication services are provided by GSI [9] in our prototype implementation.

In our implementation, users entering the Discover collaboratory using the portal are assigned a set of roles when they log in. A *Role State Machine* is then locally set up for each user, which dynamically adjusts the active role based on events from the local context agent. Similarly, the *Permission State Machines* are set up at the application (or service/resource) for each role that will access it. The *Permission State Machines* similarly adjust the active permissions based on events from the local context agent. The context agents are authorized by the central authority using GSI delegation mechanisms. The access control policy is stored in the policy repository, which

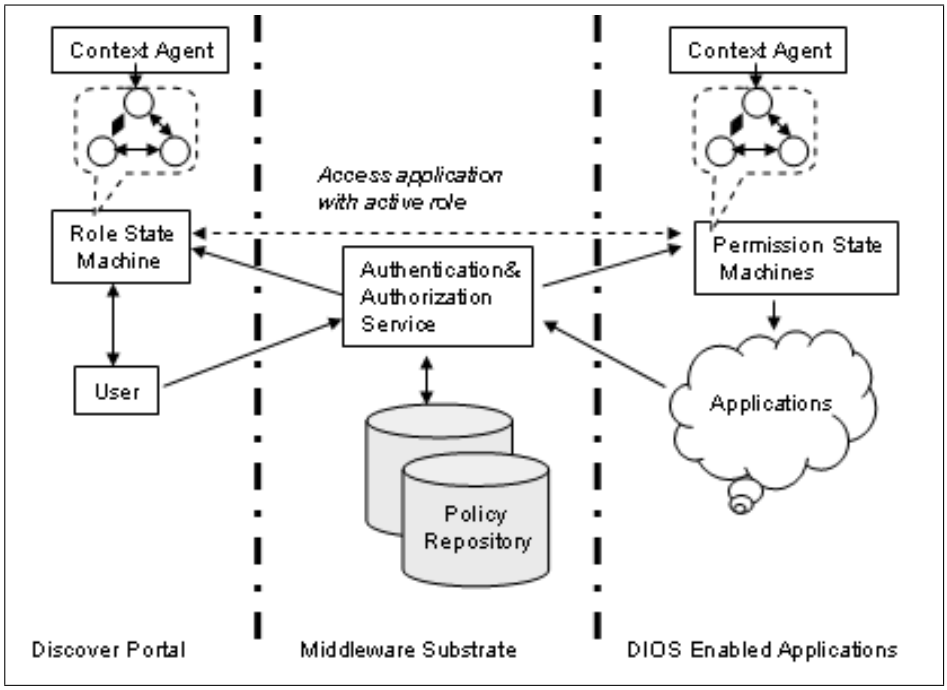

Figure 4.2: Dynamic access control in discover

is maintained by an *Authentication & Authorization Service* within *Discover Middleware Substrate*. Polices are specified in XML and define role/permission assignments and transitions as illustrated in Figure 4.3. Policies defined for our implementation include *UserPolicy*, *RoleHierarchyPolicy*, *RoleAssignmentPolicy*, *PermissionAssignmentPolicy*, *EventPolicy*, *RoleTransitionPolicy* and *PermissionTransitionPolicy*.

- UserPolicy - this policy specifies the users that are authorized to access resources,applications and services covered by the policy.
- RoleHierarchyPolicy - this policy specifies the different roles used in Discover and their relationships to each other.
- RoleAssignmentPolicy - this policy specifies the subset of roles that are assigned to the users authorized to access resources, applications and services.
- PermissionAssignmentPolicy - this policy specifies the subset of permissions that are assigned to the roles that are defined in Discover.

```
<ROLE_TRANSITION>
    <POLICY>
        <SUBJECTID>gszhang</SUBJECTID>
        <BEGIN_ROLE>Super User</BEGIN_ROLE>
        <EVENT>Unsecure Link</EVENT>
        <END_ROLE>General User</END_ROLE>
    </POLICY>

</ROLE_TRANSITION>
```

Figure 4.3: Sample RoleTransition policy in XML

- EventPolicy - this policy specifies the context information that will trigger the transition of the active role or active permission.
- RoleTransitionPolicy - this policy specifies the transition from one role to another role that is triggered by the context information.
- PermissionTransPolicy - this policy specifies the transition from one permission to another permission that is triggered by the context information.

In our prototype implementation, we assume that a security administrator will guarantee the correctness of a policy for a object or subject - i.e. SESAME sets up the *Role State Machines* and *Permission State Machines* without considering checking them for errors or conflicts. There are no inherent constraints on the number of roles and permissions, or on the relationships betweens the roles or permissions. To illustrate our implementation, consider a simple example with a single user with three roles and a Grid resource with three permissions, as shown in Table 4.1 and Table 4.2 respectively. The role and permission hierarchies for this example are shown in Figure 4.4.

Table 4.1: Permission assignments for the example.

Role	Permissions
Super User	P_1, P_2, P_3
Basic User	P_2, P_3
Guest	P_3

Table 4.2: Permission definition for the example.

Permission	Privileges
P_1	Steer Object, View Object, Basic
P_2	View Object, Basic
P_3	Basic

We consider two types of context information in our implementation: (1) Object context such as a user's location, time, local resource state and link state, and (2) Subject context, such as the current load, availability, connectivity for a resource. Context agents build on existing Grid middleware services. For example object context can be collected using the Context Toolkit [2] and subject context can be obtained using NWS [26].

4.2 Prototype Operation in Discover

The operation of the prototype is illustrated using a set of simple scenarios. These scenarios, although somewhat contrived, demonstrate the effectiveness and utility of the DRBAC model for Grid applications. For each of these scenarios, consider a user (say N) equipped with a mobile devices such as a PDA, and involved in collaboration scientific investigation using Discover. Assume that the user's environment is part of the pervasive Grid environment with appropriate middleware services.

Assume that user N logs into the system using her PDA. Based on her credentials, the *Authentication & Authorization service* assigns her a set of roles. The *Authority Service* also sets up an access control agent on her PDA, which maintains the role state machine. A DRBAC policy defined to select an appropriate role based on the level of security of her wireless connection, i.e. her active role is *Super User* while the

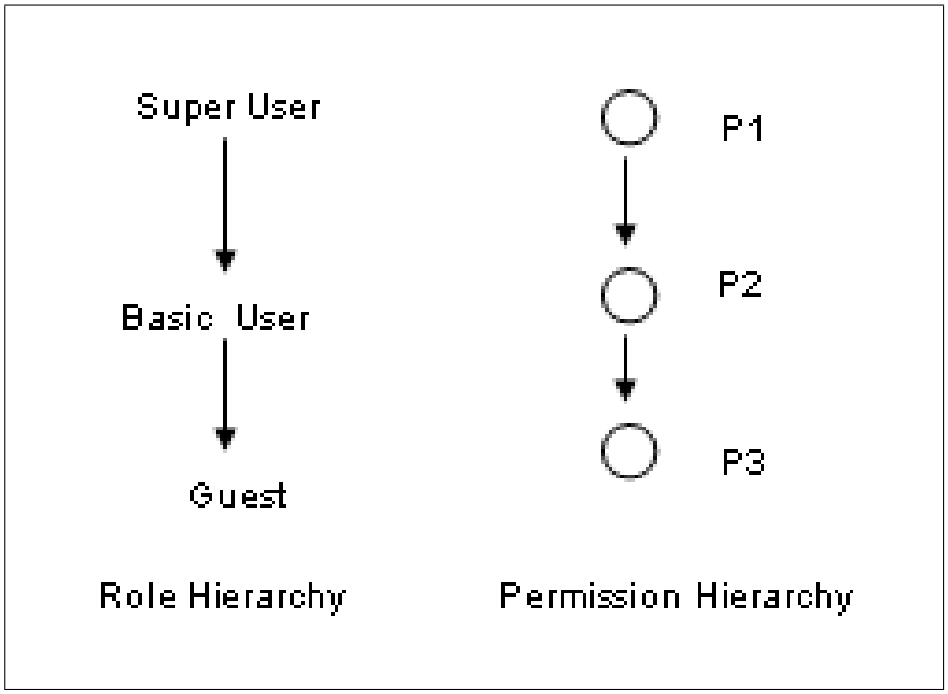

Figure 4.4: Role and permission hierarchies for the example.

network is secure (e.g. in her laboratory or office) and is *Basic User* if it is insecure. The corresponding *EventPolicy* and *RoleTransitionPolicy* may be defined as follow:

- *EventPolicy* - Generate event *insecure* when N's link has no encryption.
- *RoleTransitionPolicy* - Transit role from *Super User* to *Basic User* when event *insecure* is generated.

A corresponding permission state machine is maintained on the application side as shown in Figure 4.5. As seen in the figure each role has its own permission state machine. The dashed circle represents the current active permission for each role. A DRBAC policy is defined so that the active permission of the role *Super User* is P_1 while load is low and P_2 when the system load increases above some threshold, as there is a possibility that the application may get corrupted. The corresponding *EventPolicy* and *PermissionTransitionPolicy* may be defined as follow:

- *EventPolicy* - Generate event *high load* when load increases above *Threshold*.

- *PermissionTransitionPolicy* - Transit permission from P_1 to P_2 when event *high-load* is generated.

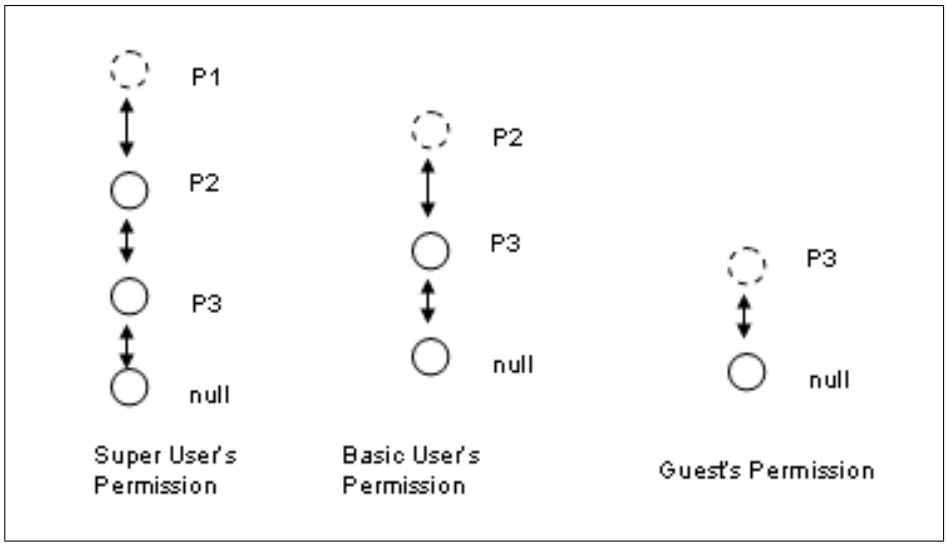

Figure 4.5: Permission hierarchy for the application

Based on the policies defined above, the following scenarios illustrate the operation of the SESAME DRBAC model.

- When user N moves out of her laboratory, the context agent will detect (using middleware context services) that the wireless network no longer has the level of encryption required and will generate the *insecure* event. This event will trigger a transition in the role state machine and downgrade her active role to *Basic user*. As a result of this transition, N will not be able to control and steer applications as she did while in her laboratory. When she reaches her office where the network is once again secure, the agent will detect this and will once again make *Super User* the active role.

- While in her office, N's active role is *Super User* and she can monitor, interact with and steer applications under normal circumstances (load at the application server is low). However if the load on the application server increases as more users join the session, the local agent generates the *highload* event, which triggers

a transition in the permission state machine and change from P_1 to P_2. As a result *Super User* will no longer be able to steer the application.

A screen dump from the *Discover Portal* during these scenarios is illustrated in Figure 4.6. As shown in this figure, due to the transitions, the portal displays "You don't have the permission to access ....". Note that for these scenarios and the experiments presented in the following section, context information was simulated.

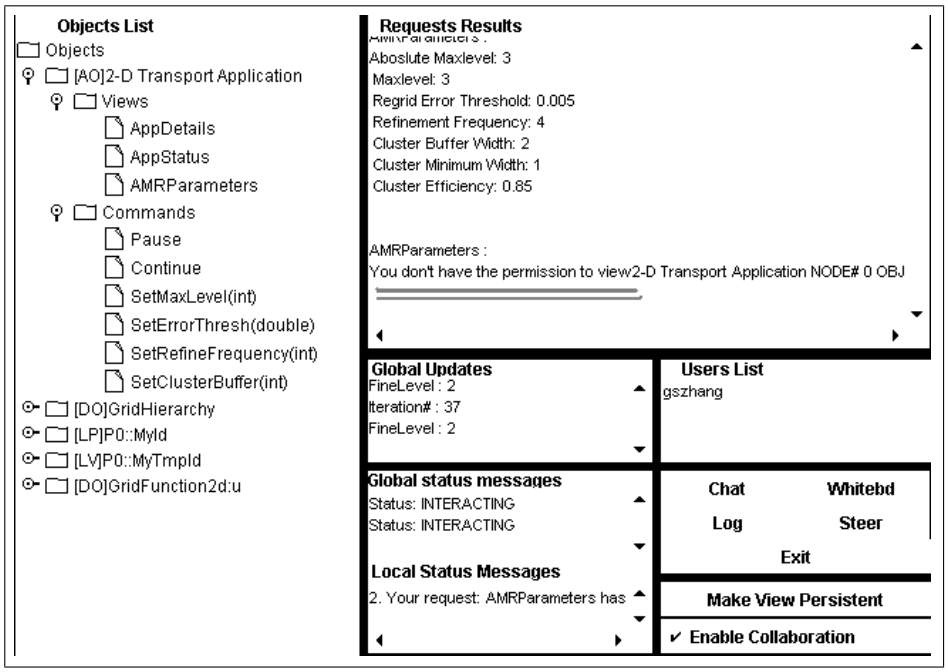

Figure 4.6: Dynamic access control in discover

In our current implementation of the DRBAC model, the active role of the user and the active permission of the role change independently. As a result, it is possible that even though the active role of user has been changed to match the current context, the user has certain permission(s) based on the previous role. We are currently addressing this potential consistency issue.

Chapter 5
Experimental Evaluation

We use the prototype implementation of SESAME in Discover to measure the overheads of the DRBAC model. The experiments were conducted on two PC using PII-200MHZ processors, running Windows NT 4.0, and one PC using PIII-500MHZ processor, running RedHat Linux 7.2. The machines were connected by a 100 Mb Ethernet switch. The *Discover Middleware* was installed on the machines running Windows NT 4.0, while the *Application* was installed on the machine running RedHat Linux 7.2. The *Discover portal* ran on the other machine running Windows NT 4.0. The following factors affect overhead of the DRBAC model.

- The number of roles assigned to the object.
- The frequency of the events (generated by the context agent at the object) that trigger transitions in the role state machine.
- The number of permissions assigned to each role.
- The frequency of the events (generated by the context agent at the subject) that trigger transitions in the permission state machine.

In the first set of experiments, we assigned each user 5 roles, and the role with highest privileges had 5 permissions. The events that triggered transitions in the role state machine were generated at different time interval. The times required to generate a request at the *Discover Portal* and get a response from the *Applications*, i.e. the interaction times, for different event frequencies are listed in Table 5.1. The first row is for the case without DRBAC.

Table 5.1: Interaction time in ms. for different context event frequencies.

Event frequency	Time (ms.)
-	2300
1min	4732
2min	4403
3min	4102
4min	3482
5min	3104

In the second set of experiments, we randomly generate events to trigger transitions in the role state machine and vary the number of roles assigned. The role with the highest privileges is still assigned 5 permissions. Table 5.2 shows the interaction times for different number of roles.

Table 5.2: Interaction time in ms. for different number of roles.

Number of Roles	Time (ms.)
-	2300
5	2520
6	2608
7	2804
8	2920
9	3004

In the last set of experiments, the user had a state machine with 5 roles and the role with the highest privileges was set as the active role. Events were randomly generated at the application server to trigger transitions in the permission state machine. The number of permissions assigned to the active role was varied. The interaction times for different number of permissions are listed in Table 5.3.

Table 5.3: Interaction time in ms. for different number of permissions.

Number of Permissions	Time (ms.)
-	2300
5	2500
6	2602
7	2698
8	2804
9	2912

These preliminary results show that in general the overheads of the DRBAC implementation are reasonable. The primary overheads were due to the event generated by the context agent - the higher the frequency, the larger was the overhead. The context agent can be implemented as an independent thread and as a result, the transition overheads at the object and subject are not significant.

Chapter 6
Summary and Conclusions

6.1 Discussion

The major strength of our DRBAC model is its ability to make access control decision dynamically according to the context information. Its dynamic property is particularly useful for the pervasive Grid applications. Obviously, our design will make the system complex. To successfully implement our model into the real applications, some issues should be took into consideration:

- As we use the context information as a key player while grant access privilege, we must guarantee the security of the context information. Compromised context information will let the system make wrong access control decision.

- In our model, the active role of the user and the active permission of the role will change dynamically. It is possible that in some situation the active role of user has been changed but the user has already got certain permission to access the resource with the old role. We need some mechanism to keep the consistency.

- To implement dynamic access control in real system, we need utilize some context toolkit to collect the context information of both the user and system. The overhead of such kind of services should be considered carefully.

- Because the role state machine will run on the user's equipment(PDA, mobile phone), the resource consumption of the mobile terminal will increase. however,

the users generally have 3-10 roles assigned to them when they are involved in different pervasive Grid applications. To maintain a state machine with 3-10 states has little if any affect on the performance of the user's equipment compare with the dramatic increasing power of the mobile device.

- We assume that a central authority will maintain the role hierarchy and permission hierarchy for all the user and resource in the system. This is feasible for most of current pervasive Grid applications. where the amount of the user and resource is not huge. To deploy our model into more distributed system, the scalability issue need to be addressed with certain mechanism.

6.2 Conclusion

In this thesis, we presented the SESAME dynamic context-aware access control mechanism for pervasive Grid applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context. The underlying dynamic role based access control (DRBAC) model extends the classic role based access control (RBAC). A prototype implementation of SESAME and the DRBAC model within the Discover computational collaboratory was presented. The feasibility, performance and overheads of SESAME were experimentally evaluated. The results show that the overheads of the model are reasonable and the model can be effectively used for dynamic context-aware access control for Grid applications.

6.3 Future Work

In this thesis, we introduced an new access control model, Dynamic Role Based Access Control(DRBAC), and with an example application,we explain why our model will be useful to secure the pervasive Grid applications. However, access control alone can

not provide security, our DRBAC must combine with some feasible authentication mechanisms to secure the pervasive Grid applications in the real world.

In our approach, we don't mention delegation, which is proved to be important for pervasive Grid applications. We note the work of Barka [3] , who gives a framework for Role-Based Delegation Models. Lalana Kagal [12] described a delegation architecture for pervasive computing. We will continue work to include delegation into our model.

Appendix A
Role Based Access Control Models

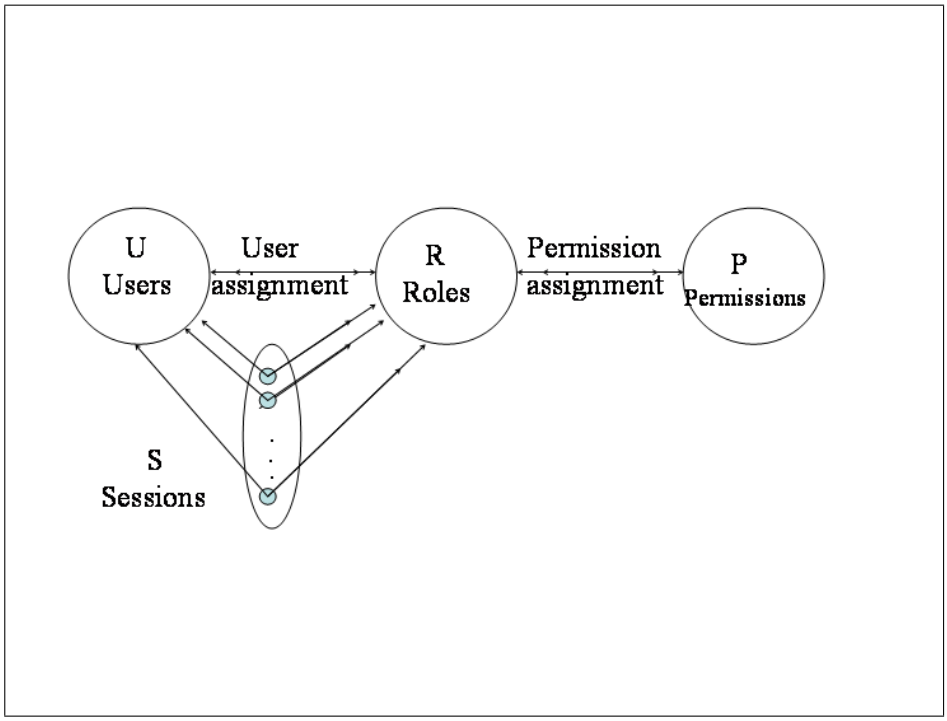

Figure A.1: Role Based Access Control Model: RBAC0

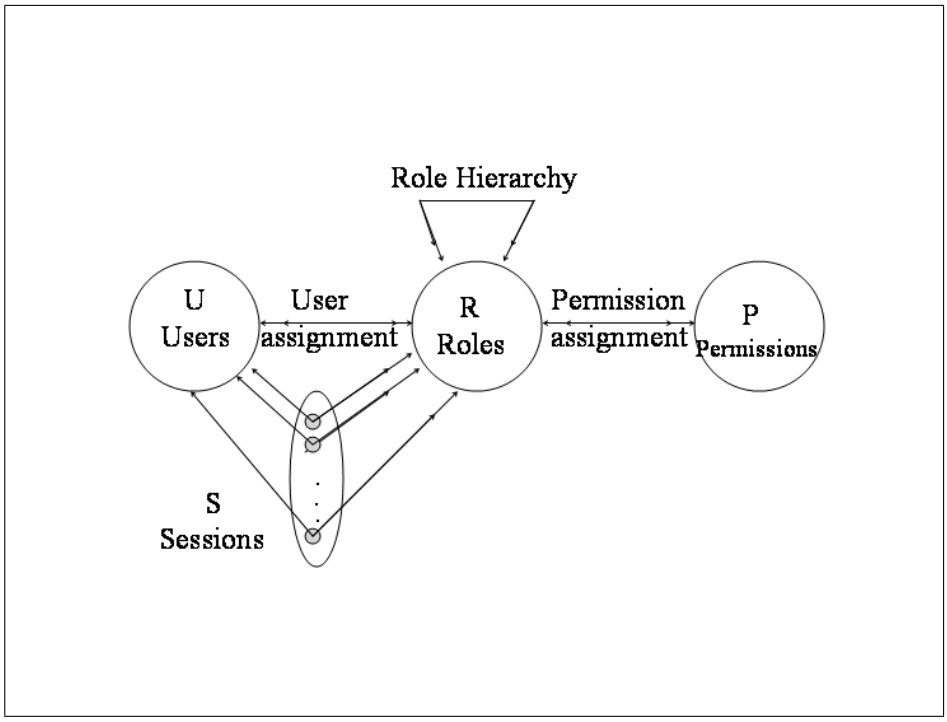

Figure A.2: Role Based Access Control Model: RBAC1

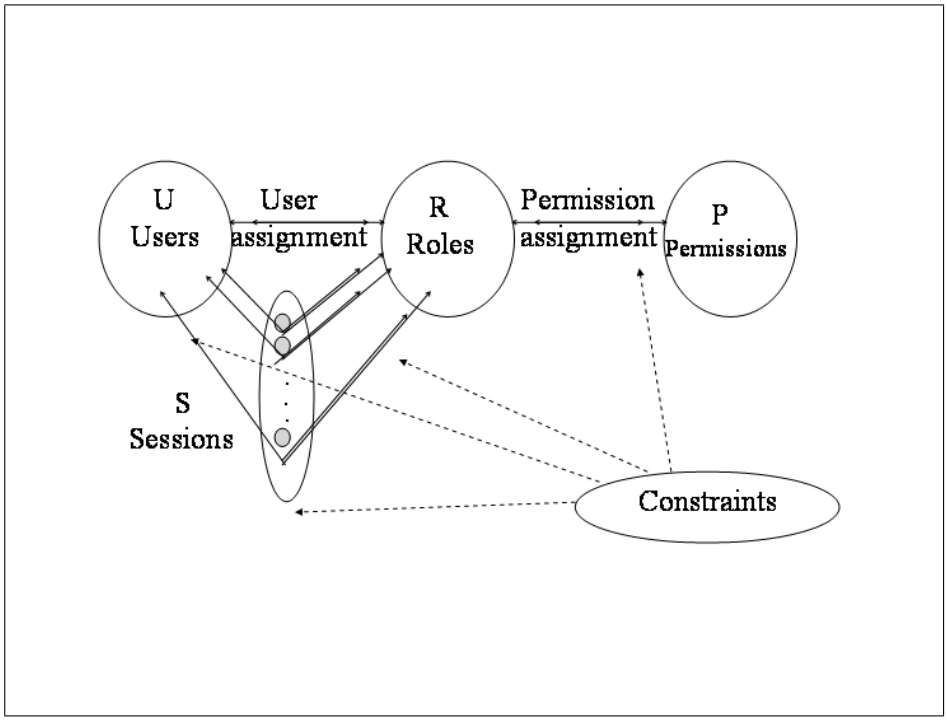

Figure A.3: Role Based Access Control Model: RBAC2

References

[1] T. A. Corbi A. G. Ganek. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1), 2003.

[2] G. D. Abowd A. K. Dey. The context toolkit: Aiding the development of context-aware applications. In ACM Press, editor, *Human Factors in Computing Systems: CHI 99*, pages 434–441, Pittsburgh, PA, USA, May 1999.

[3] E. Barka and R. Sandhu. Framework for role-based delegation models. In *16th Annual Computer Security Applications Conference (ACSAC'00)*, New Orleans,Louisiana,USA, 2000.

[4] V. Bhat and M. Parashar. A middleware substrate for integrating services on the grid. Technical Report TR-268, Center for Advanced Information Processing, Rutgers University, November 2002.

[5] J. F. Barkley D. F. Ferraiolo and D. R. Kuhn. A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1):34–64, 1999.

[6] S. Gavrila D. R. Kuhn D. F. Ferraiolo, R. Sandhu and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[7] Global Grid Form. Global Grid Form Web Site, 2003. http://www.ggf.org/.

[8] C. Kesselman I. Foster and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15(3):200–222, 2001.

[9] G. Tsudik S. Tuecke I. Foster, C. Kesselman. A security architecture for computational grids. In *5th ACM Conference on Computer and Communications Security Conference*, pages 88–92, San Francisco, CA, USA, 1998.

[10] K. Beznosov J. Barkley and J. Uppal. Supporting relationships in access control using role based access control.

[11] V. Welch K. Keahey. Fine-grain authorization for resource management in the grid environment. In *3rd International Workshop on Grid Computing - Grid 2002*, Baltimore, MD, USA, 2002.

[12] T. Finin L. Kagal and A. Joshi. Trust-based security in pervasive computing environments. *IEEE Computer*, 34(12):154–157, 2001.

[13] I. Foster C. Kesselman L. Pearlman, V. Welch and S. Tuecke. A community authorization service for group collaboration. In *the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, CA, USA, 2002.

[14] J. Alameda R. Ananthakrishnan M. Govindaraju A. Slominski K. Connelly V. Welch D. Gannon R. Bramley L. Ramakrishnan, H. Rehn and S.Hampton. An authorization framework for a grid based component architecture. In Springer Press, editor, *3rd International Workshop on Grid Computing*, pages 169–180, Baltimore, MD, USA, November 2002.

[15] H. Liu V. Matossian V. Putty C. Schmidt G. Zhang L. Zhen M. Agarwal, V. Bhat and M. Parashar. Automate: Enabling autonomic grid applications. In *the Autonomic Computing Workshop, 5th Annual International Active Middleware Services Workshop (AMS2003)*, Seattle, WA, USA, June 2003.

[16] M. J. Moyer M. J. Covington and M. Ahamad. Generalized role-based access control for securing future applications. In *23rd National Information Systems Security Conference. (NISSC 2000)*, Baltimore, Md, USA, October 2000.

[17] S. Srinivasan A. Dey M. Ahamad M. J. Covington, W. Long and G. Abowd. Securing context-aware applications using environment roles. In *6th ACM Symposium on Access Control Models and Technologies (SACMAT '01)*, Chantilly, Virginia, USA, May 2001.

[18] Z. Zhan M. J. Covington, P. Fogla and M. Ahamad. A context-aware security architecture for emerging applications. In *the Annual Computer Security Applications Conference (ACSAC)*, Las Vegas, Nevada, USA, December 2002.

[19] D. Kafura M. Lorch. Supporting secure ad-hoc user collaboration in grid environments. In Springer Press, editor, *3rd International Workshop on Grid Computing*, pages 181–193, Baltimore, MD, USA, November 2002.

[20] V. Mann and M. Parashar. Engineering an interoperable computational collaboratory on the grid. *Special Issue on Grid Computing Environments, Concurrency and Computation: Practice and Experience*, 14(13-15):1569–1593, 2002.

[21] R. Muralidhar and M. Parashar. A distributed object infrastructure for interaction and steering. *In Concurrency and Computation: Practice and Experience*, to appear.

[22] Globus Project. Globus Project Web Site, 2003. http://www.globus.org/.

[23] I. Foster C. Kesselman S. Tuecke J. Volmer V. Welch R. Butler, D. Engert. A national-scale authentication infrastructure. *IEEE Computer*, 33(12):60–66, 2000.

[24] D.Ferraiolo R. Sandhu and R. Kuhn. The nist model for role-based access control: Towards a unified standard. In *5th ACM Workshop on Role Based Access Control*, pages 47–64, Berlin, Germany, 2000.

[25] H. Feinstein R. Sandhu, E. Coyne and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[26] Network Weather Service. University of California, Santa Barbara, Research Project Web Site, 2003. http://nws.cs.ucsb.edu/.

[27] R. Muralidhar V. Mann, V. Matossian and M. Parashar. Discover: An environment for web-based interaction and steering of high-performance scientific applications. *Concurrency and Computation: Practice and Experience*, 13(8-9):737–754, 2001.

[28] S. Mudumbai W. Johnston and M. Thompson. Authorization and attribute certificates for widely distributed access control. In *IEEE 7th International Workshops on Enabling Technologies: Infrastructures for Collaborataive Enterprises*, Stanford University, CA, USA, 1998.

[29] G. Zhang and M. Parashar. Dynamic context-aware access control for grid applications. In IEEE Computer Society Press, editor, *4th International Workshop on Grid Computing (Grid 2003)*, pages 101 – 108, Phoenix, AZ, USA.

[30] G. Zhang and M. Parashar. Context-aware dynamic access control for pervasive computing. In *2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04)*, San Diego, California, USA, January 2004.