

UNIVERSITY OF CALGARY

Dynamic Role Lease Authorization Protocol
for a Distributed Computing System

by

Nelson Chung Ngok CHU

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

January, 2013

© Nelson Chung Ngok CHU 2013

UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled “Dynamic Role Lease Authorization Protocol for a Distributed Computing System” submitted by Nelson Chung Ngok Chu in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Supervisor, Dr. Kenneth E. Barker
Dean, Faculty of Science

Dr. Reda S. Alhadj
Professor, Department of Computer Science

Dr. Zongpeng Li
Associate Professor, Department of Computer Science

Dr. Behrouz Far
Professor, Schulich School of Engineering

Dr. Polly Huang
Professor, Department of Electrical Engineering
National Taiwan University

Date: December 17, 2012

Abstract

A distributed computing system, such as a Grid, could be a very dynamic environment and the user groups are most likely become highly diverse. A user group could be formed by the users of different networks, organizations, or administrative-domains with different hardware/software infrastructures and managerial policies. Handling requests from a wide range of users from different domains becomes a challenge when attempting to accommodate all the differences. Service providers find it impossible to track all users (the number of users could be potentially very large) in a Grid. Therefore, an access control mechanism that provides users appropriate access to the resources in a dynamic environment is required. Role Based Access Control (RBAC) models have been demonstrated to be an effective and efficient approach for an administrator to manage accesses in a computing system. Much has been done to adapt the RBAC concept to Grids and focus on the authorization and verification of the dynamic factors or contexts of a user, such as time, location, rank, *etc.* Some applications also allow administrators to change the policies during the authorization process, but they did not handle the authorization in real-time and on-demand manner in a Grid. It is a critical authorization requirement for a dynamic environment. Therefore, this problem motivated us to develop a new dynamic authorization protocol, Dynamic Role Lease Authorization (DRLA) that is suitable for a dynamic distributed computing environment.

Acknowledgements

This dissertation would not have been possible without the guidance and the help of several individuals who, in one way or another, contributed and extended their valuable assistance in the preparation and completion of this research.

First and foremost, my utmost gratitude to my supervisor, Dr. Ken Barker, whose guidance and encouragement I will never forget. Dr. Barker has been my inspiration as I hurdled all the obstacles to the completion this research. Ken has not only trained me to be a researcher, but also taught me the integrity of a leader that has become an invaluable asset for my life.

I would like to thank Dr. Reda Alhajj for his guidance and encouragement during my graduate study. I am also thankful to Dr. Rob Simmons from the WestGrid and Dr. Polly Huang from the National Taiwan University. Their guidance helped me to define some essential concepts in this research.

Lastly, I would like to thank my parents Chi-wan Yau and Ki-Kit Chu, my brothers Chris and Vincent, my daughter Nicole, and my wife Ying. Without their encouragement, support and patient, this work would not have been possible.

*To my lovely wife, Ying,
who fills my life with love and support*



*To my lovely parents, Chi-wan & Ki-kit,
who make my life shine*

TABLE OF CONTENTS

ABSTRACT	III
ACKNOWLEDGEMENTS.....	IV
LIST OF FIGURES	VIII
CHAPTER 1: INTRODUCTION	1
1.1. MOTIVATION	1
1.2. DYNAMIC ACCESS CONTROL REQUIREMENTS IN A DISTRIBUTED ENVIRONMENT	2
1.3. PROBLEM STATEMENT	4
1.4. CONTRIBUTIONS	6
1.5. ORGANIZATION OF THE THESIS.....	7
CHAPTER 2: BACKGROUND.....	8
2.1. GRID COMPUTING.....	9
2.2. CLOUD COMPUTING	11
2.3. GRID VS CLOUD	12
2.4. SECURITY TECHNOLOGIES AND ATTACKS.....	15
I) TRANSPORT LAYER SECURITY (TLS) / SOCKETS LAYER (SSL) PROTOCOLS	16
II) VIRTUAL PRIVATE NETWORK (VPN).....	17
III) DENIAL-OF-SERVICE (DOS) ATTACK.....	18
IV) DNS CACHE POISONING ATTACK.....	19
V) MAN-IN-THE-MIDDLE (MITM) ATTACK.....	20
VI) SSL MITM ATTACK	21
VII) VPN VULNERABILITY	22
2.5. ACCESS CONTROL MODEL [8]	24
I) PRINCIPLE OF LEAST PRIVILEGE	24
II) DISCRETIONARY ACCESS CONTROL (DAC)	26
III) MANDATORY ACCESS CONTROL (MAC).....	26
IV) ROLE BASED ACCESS CONTROL (RBAC)	26
V) DYNAMIC CONTEXT AWARE AUTHORIZATION MODELS	28
A) CONTEXT-AWARE ACCESS CONTROL	29
B) DYNAMIC AUTHORIZED ROLE BASED ACCESS CONTROL	29
C) DYNAMIC AUTHORIZATION FRAMEWORK FOR MULTIPLE AUTHORIZATION TYPES.....	30
D) EXTENDED USAGE CONTROL.....	31
E) DYNAMIC AUTHORIZATION MANAGEMENT MODEL	32
VI) ACTIVE/DYNAMIC AUTHORIZATION MODELS	33
A) DYNAMIC ROLE BASED ACCESS CONTROL.....	33
B) ACTIVE AUTHORIZATION MODEL FOR MULTI-DOMAIN COOPERATION.....	35
C) ON-WHEN-THEN-ELSE EVENT-BASED AUTHORIZATION.....	36
D) TASK-BASED ACCESS CONTROL	37
E) DYNAMICALLY ADMINISTRATED ROLE-BASED ACCESS CONTROL	39
2.6. LEASE	41
I) DHCP LEASE "LIFE CYCLE"	47
II) JINI	47
III) GLOBUS TOOLKIT - LEASE-BASED LIFETIME MANAGEMENT	48
IV) REMOTING LIFETIME AND LEASES IN .NET	48
2.7. SUMMARY	49

CHAPTER 3: DYNAMIC ROLE LEASE AUTHORIZATION PROTOCOL	50
3.1. INTRODUCTION.....	50
3.2. BASIC GRID ENVIRONMENT.....	55
I) SCENARIO 1: FULLY CONNECTED ERROR-FREE ENVIRONMENT	55
II) SCENARIO 2: PARTIALLY CONNECTED ERROR-FREE ENVIRONMENT.....	60
III) SCENARIO 3: ERROR HANDLING IMPLEMENTED ENVIRONMENT	62
3.3. DYNAMIC ROLE LEASE AUTHORIZATION (DRLA) PROTOCOL.....	64
I) PHYSICAL STRUCTURE.....	68
II) DRLA AUTHORIZATION PROCESS.....	72
III) USE CASE.....	80
3.4. SUMMARY	83
CHAPTER 4: SIMULATION WITH DRLA PROTOCOL	84
4.1. SIMULATION ENVIRONMENT	84
4.2. SIMULATION DETAILS.....	85
4.3. TEST CASES	90
4.4. SUMMARY	93
CHAPTER 5: SIMULATION RESULTS AND ANALYSIS	94
5.1. ANALYSIS ON NETWORK TOPOLOGY DIFFERENCE.....	97
5.2. ANALYSIS ON NON-ERROR-FREE ENVIRONMENT.....	99
5.3. ANALYSIS ON DYNAMIC USER CONTEXT & POLICY CHANGES	102
5.4. ANALYSIS WITH OTHER AUTHORIZATION MODELS	105
5.5. SUMMARY	114
CHAPTER 6: CONCLUSION AND FUTURE WORK	115
6.1. CONTRIBUTIONS	116
6.2. RESEARCH DIRECTION	118
BIBLIOGRAPHY	122

LIST OF FIGURES

Figure 2-1: Cloud Computing Service Model	11
Figure 2-2: Distributed Computing Systems	14
Figure 2-3: TLS and SSL	16
Figure 2-4: Virtual Private Network.....	17
Figure 2-5: Denial-of-Service (DoS).....	18
Figure 2-6: DNS Cache Poisoning	19
Figure 2-7: Man-In-The-Middle (MITM)	20
Figure 2-8: SSL MITM attack between HTTP and HTTPS.....	21
Figure 2-9: MITM attack on SSL certificate	22
Figure 2-10: VPN Vulnerabilities	23
Figure 2-11: Role Based Access Control (RBAC) [1]	28
Figure 2-12: Context-Aware Access Control [6]	29
Figure 2-13: Extended Usage Control [17]	31
Figure 2-14: Dynamic Authorization Management Model [43].....	32
Figure 2-15: Dynamic Role Based Access Control Model [8, 7].....	34
Figure 2-16: Active Authorization Management Model [32].....	35
Figure 2-17: On-When-Then-Else Event-Based Authorization (OWTE) [33].....	36
Figure 2-18: Task-Based Access Control (TBAC) [36].....	38
Figure 2-19: Dynamically Administrated Role-Based Access Control [34, 35]	40
Figure 3-1: Access control system with “on/off” approach.....	52
Figure 3-2: DRLA access control system.....	53
Figure 3-3: Fully connected error free environment.....	57
Figure 3-4: Partially connected error free environment	60
Figure 3-5: Job distribution lists.....	61
Figure 3-6: Conceptual Diagram of DRLA	64
Figure 3-7: Relationship Diagram of DRLA Components	65
Figure 3-8: DRLA-enabled Grid	70
Figure 3-9: DRLA grid service.....	74
Figure 3-10: DRLA authentication and algorithms	75
Figure 3-11: DRLA authorization and algorithms.....	77
Figure 3-12: DRLA authentication enforcement and algorithms	79
Figure 4-1: Flow Diagram of a Request in the Simulated Environment	87
Figure 4-2: DRLA grid service.....	88

Figure 4-3: Fully connected network in the simulated environment	90
Figure 4-4: Partial connected network in the simulated environment	91
Figure 5-1: A Simple of the simulation Output	96
Figure 5-2: Total Simulation Time in an Error-free Environment	97
Figure 5-3: DRLA Protocol Total Execution Time in an Error-free Environment	98
Figure 5-4: DRLA Protocol Average Execution Time in an Error-free Environment	99
Figure 5-5: Total Simulation Time in a non-error-free Environment.....	100
Figure 5-6: DRLA Protocol Execution Time in a Non-error-free Environment	101
Figure 5-7: DRLA Protocol Ave. Execution Time in a Non-error-free Environment.....	101
Figure 5-8: Users' request and policy for contextual change testing.....	101
Figure 5-9: Sample output – request submission.....	103
Figure 5-10: Sample output – authorization result for the first request	103
Figure 5-11: Sample output – authorization result for the second request	103
Figure 5-12: Sample Users' request and policy for policy change testing	104
Figure 5-13: Sample output – Policy set #1	104
Figure 5-14: Sample output – Policy set #2	104
Figure 5-15: DRLA Experiment without Role Change	106
Figure 5-16: DRLA Experiment with Role Change	107
Figure 5-17: DRLA Experiment with Context Change	108
Figure 5-18: CAAC Implementation	109
Figure 5-19: DRBAC Implementation	110
Figure 6-1: Grid Data Access Management with DRLA and Tuple Spaces	119

Chapter 1: INTRODUCTION

1.1. Motivation

Grid computing has become one of the main paradigms of distributed computing systems in the last decade. It provides a framework that allows Grid users to share, collaborate, and access physically distributed resources, including hardware, software, and information, to accomplish resource-intensive tasks over the network across multiple virtual organizations (VOs). In the Grid community, there has been a significant research effort put forth to study, standardize, develop, and deploy their architectures, protocols, and applications. These three components cover many research areas, including security (authentication, authorization, access control, *etc.*); resource monitoring and discovery; resource/task scheduling; data management (database design, data integration, data transfer, *etc.*), software packaging and distribution; and so on. However, these developments are still facing many challenges presented by the Grid infrastructure. These difficulties are mostly caused by the heterogeneities between different VOs; policy differences in different domains; and the highly dynamic nature of a Grid environment [8]. As expected by the Grid community, the Grid has become a rapidly emerging infrastructure, and the number of Grid users is also expected to increase quickly. On the one hand, we should provide a flexible mechanism to allow the users to access the resources. On the other hand, we have to eliminate any unauthorized access and minimize any security risks in the system. Therefore, one critical challenge is to provide a flexible authorization and access control mechanism that is able to minimize any unauthorized access to the system. This has motivated us to focus on a new dynamic authorization approach that is suitable for a dynamic distributed computing environment.

1.2. Dynamic Access Control Requirements in a Distributed Environment

In a large distributed computing environment, such as a Grid, most interactions and collaborations involve many different components in the system, such as users, hardware and software, access policies, business rules, *etc.* The dynamic factors in a distributed computing environment affect the interactions and collaborations between the components. A dynamic factor is defined as any component and its attributes (such as user participation pattern, user contextual information, heterogeneity between systems, temporal constraint, *etc.*) that may change its behaviors and status. To provide effective and efficient access control services, we must understand the dynamic-relationship between these components.

A Grid is a federation of multiple, individually administrated computing domains; it is referred to as virtual organizations (VOs). In a VO, there are two groups of elements: users and resources. Users include human or automated non-human agents, including, software agent and computerized robot. Resources include all other components (hardware, software, and information) used by users, such as physical computer network, data storage, or computer. Most activities in a Grid are the collaboration within one of them or between these two groups in one or more VOs. Each VO has its own access control policies and could be changed as needed to fit its own situation and requirements. For example, an international scientific research project may involve researchers and computational resources from different universities and companies in different countries. These users (researchers) and resources (computers, disk, *etc.*) belong to different VOs (universities and companies). Obviously, different universities and companies are in different administrative domains and have different local and/or global access control policies. Beyond these settings, there are many other dynamic factors that may require

changes to existing policies. For instance, a user may move from a trusted network to a non-trusted network with a different set of policies and security requirements. To maintain the security level on the resource provider's system, the resource provider may want to revoke the user's access to prevent any potential security risk. This implies that an access control system should consider the dynamic changes of a user or resource to grant required permission. In other words, the access control in a dynamic environment should be updated dynamically.

Another key requirement in a dynamic access control system is to be context aware. Context is not limited to some physical parameters (such as location, network, and time), but also includes business parameters/rules (such as priority, trust, or load balance management.) For example, user A may have higher priority than other users, so the granted access to others should be modified and permit user A to use the resources first even though user A may be a late comer. However, this should be handled very carefully with user's context and policy management, otherwise resource livelock may occur. In a lease based authorization, like the one in this thesis, this kind of problem can be handled by setting a lease period or timeout. Another situation that may require dynamic access control: resource A may trust the users from domain B but not domain C. Subsequently, domain B is merged with domain C and form a new domain, D. In most trust model, domain D becomes not trusted. In this situation, resource A may refuse or reconsider (based on the defined policy) the access request from the old users in domain B or the new domain D. Therefore, being context aware, including physical and business parameters, becomes a critical requirement.

In many current access control systems, a single sign-on authorization may be implemented to provide a more convenient service. However, the single-on authorization is taking a “turn on/off” approach [36]; the permission is granted once the authorized user signs on. This approach may not be appropriate in a Grid environment due to many dynamic factors that may raise potential security risks, such as unauthorized access from a non-trusted network. It is unsuitable for a dynamic distributed environment since the “turned on” permission may be left behind when the user is disconnected from the system due to any unexpected problem; it thereby adds extra risk to the system. Simply applying a different set of permissions may not resolve the problems because the dynamic factors are not limited in the cross-domain interaction and collaborations. As suggested by the “Principle of Least Privilege” [2, 8, 41], any permission should be granted as needed to perform the required tasks. Therefore, minimizing any long-held permission by any user is recommended, and most permission should be granted and assigned on a demand basis [2, 8, 41]. The dynamic factors should be considered in the authorization processes. Therefore, a new authorization is needed to handle the factors and manage the access dynamically; this problem is the main focus of this thesis.

1.3. Problem Statement

A distributed computing system, such as a Grid, could be a very dynamic environment. It is impossible for each service provider to track all Grid users (potentially very large) with their constantly changing contextual information and the non-static resource policies. However, these dynamic users’ contextual information could be the keys of an authorization decision in a Grid. Therefore, a new access control mechanism is required to provide a user with appropriate access to the resources in this dynamic environment

without tracking the users and policies. Role Based Access Control (RBAC) models have been demonstrated to be an effective and efficient approach for an administrator to manage access in a computing system [1, 2]. Much has been done to adapt RBAC [6, 7, 8, 9, 16, 17, 43] and dynamic authorization concepts in Grid environments. Many of them focused on validating the dynamic factors or user contexts, such as time, location, rank, and resource policies [32, 33, 34, 35, 36, 37]. However, no authorization model has been found in the reviewed literature that handles the real-time/on-demand authorization regarding the dynamic factors and dynamic changes in a Grid. Providing the right resource/information to the right users at the right time is the ultimate goal for a Grid. However, many current authorizations are a “turn on/off” process in that the permission is granted once the authorized user signs on. This mechanism is not suitable for a dynamic distributed environment since the “turned on” permission may be left behind when the user is disconnected from the system due to any dynamic problem. There are many hacking techniques that could be used to steal the left-behind permissions, thereby, gain access to the resource and/or information. As suggested in the Principle of Least Privilege, any permission should be granted as needed to perform the required tasks. To address the authorization problems caused by the dynamic factors in a Grid, a new authorization protocol, Dynamic Role Lease Authorization (DRLA) [121] is presented in this thesis. The focus of this research is to develop the dynamic authorization protocol; therefore, all other components (such as, authentication, job submission, job scheduling, *etc.*) are assumed to be already developed and providing required information for the authorization which complies to DRLA.

1.4. Contributions

Although the basic concepts (role based authorization and leasing) in DRLA are not new, but the lease based authorization protocol (DRLA) proposed in this research is effective and straightforward for addressing the dynamic authorization problem in a Grid by combining the Lease and RBAC concepts. This thesis provides contributions in three areas.

1. The proposed protocol, DRLA [121], (Section 3.3) provides a new mechanism for dynamic authorization in a distributed computing system, such as a Grid, which is highly heterogeneous and dynamic. A role leasing concept is introduced in DRLA to utilize the resources in a Grid. In such an environment, some idle resources could be assigned to a non-active or disconnected user due to some dynamic factors or failures. The role leasing structure leases out the role/permission to use a resource; the resource will be released back to the system once the lease is expired. This approach permits a resource to be occupied within the leasing period only; it implies that any failure may only hold the resource until the end of leasing period. The leasing structure minimizes the impact caused by a failure, and in the meantime, it allows a short absence or disconnection from the system that may be required in some dynamic situations. For example, a user may require a short disconnection to switch from one access point to another. The user can maintain the leased access/privilege as long as the connection is re-established within the leasing period and all policies are still satisfied. This is a common problem in a distributed system; however, a Grid environment is a highly heterogeneous and dynamic computing environment, so it

makes the problem more complicated and requires a more flexible authorization model.

2. DRLA maintains the minimum privileges that is specified in the policies for each user, and dynamically assigns and grants the required privileges to the corresponding authorized user through the lease structure (Section 3.1). The dynamic contexts and policies are verified before the access is granted. This approach satisfies the principle of least privilege and minimizes the security risk that has been identified in the “turn on/off” approach being used in many access control systems. Achieving these two aspects is essential in a Grid access control system.
3. DRLA provides on-going dynamic context aware authorization in a Grid environment (Section 3.3.ii). The authorization process in a highly dynamic computing system and should not be a one-time process; it should be an on-going procedure to ensure that users satisfy the authorization requirements all the time. Unlike many other authorization models, DRLA establishes an on-going mechanism to monitor users’ contextual information and enforce the security requirements.

1.5. Organization of the Thesis

The rest of this thesis is organized as follows: Chapter 2 provides reviews of some research about the authorization in a distributed system. Chapter 3 presents a new dynamic authorization protocol, DRLA. Chapter 4 describes the implementation of the proposed access control protocol in a simulated Grid environment. Chapter 5 discusses the simulation result and analyzes the performance of DRLA. We also compare some functionality differences between DRLA and other authorization models. Conclusions and future research directions are provided in Chapter 6.

Chapter 2: BACKGROUND

The Grid is a federation of many Grids with different computing resources, which is built with standardized methodologies and protocols to provide secure access to the dynamic networked resources and services that provide information and knowledge for an authorized user from anywhere at any time. The idea is similar to a Power Grid; a user can consume the resource anywhere at any time, regardless of the location and the ownership of the resources and services. The main goal of a Grid is to share the resources and collaborate with other users to accomplish resource-intensive tasks transparently. To accomplish this goal, there are many challenges, such as providing a secure authorization model for a wide range of users and organizations on a Grid. Another challenge is agreeing on communication protocols for the Grid. Setting up a common language regarding concepts and terms in the Grid is one of the many difficulties as well. Each Grid is an individually administrated domain and has a different set of requirements and limitations to set up the infrastructures, policies, and administrative procedures for a specific group of users in the Grid. Furthermore, the heterogeneity of the resources (including hardware, software, network and data), security policies, and protocols between each domain introduce a huge impediment to reaching a balanced agreement that satisfies each participant's interests and requirements. Most resource and service providers are trying to retain the right to control their resources as much as possible.

This dissertation focuses on the access control problems in a Grid and proposes a new authorization protocol, Dynamic Role Lease Authorization (DRLA) [121], to address the authorization concerns raised by the dynamic factors in a Grid. The remainder of this chapter will focus on the background information associated with the current state of the

related Grid technologies and problems. First, we review some basic characteristics of a Grid in Section 2.1. This research proposes a new authorization protocol that addresses some authorization problems of a Grid; therefore, having an overview (Section 2.4) of some general security technologies and problems helps us to understand the requirements for the new protocol. In Section 2.5, we discuss a more specific topic, access control, in the Grid security. We review some general access control models (such as DAC, MAC, and RBAC), and some models specifically for a dynamic distributed system. The last section of this chapter, we review the leasing concept that is adopted in the DRLA.

2.1. Grid Computing

A Grid is a system that is concerned with the integration, virtualization, and management of services and resources in a distributed heterogeneous environment. It supports collections of users and resources (a virtual organization) across various administrative and organizational domains (real organizations). A Grid computing environment combines distributed pools of resources onto which applications or services may be dynamically provisioned and re-provisioned. The contributed resources are often consolidated from numerous smaller pools, where they may have been underutilized. A Grid offers great flexibility, and it typically uses service-oriented architecture (SOA) to build the services rather than the components. The real problem that underlies the Grid concept is that it is difficult to coordinate resource sharing for solving problems in a dynamic and multi-institutional virtual organization. The “sharing” that we refer to here is not primarily file exchange but rather direct access to computers, software, data, and other resources. Direct access is required by a wide range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering.

This sharing is highly controlled by resource providers and it carefully defines just what is shared and who is allowed to share it, as well as the sharing conditions. A set of individuals and institutions defined by such sharing rules form what we call a virtual organization (VO). Grids vary tremendously in their purpose, scope, size, duration, structure, and community.

Key requirements for successful Grid implementation and management include standardization of the interfaces of common components, and the use of standardized information and data models. Grid computing is a key research area that requires addressing many challenges in different areas, such as security. There is a need for a highly flexible architecture for sharing of varied resources, ranging from programs, files, and data to computers, sensors, and networks. There is a desire for a promising resource management structure for dealing with diverse usage modes, ranging from single user to multi-user and from performance sensitive to cost-sensitive. Furthermore, the structure must embrace issues related to quality of service, scheduling, co-allocation, and accounting. There is a demand for a sophisticated security model to provide flexible control over how shared resources are used, including fine-grained and multi-stakeholder access control, delegation, and application of local and global policies.

In last couple years, Cloud computing has emerged as one of the most popular technologies, providing on-demand computing infrastructure, platform, and software as services (IaaS, PaaS, and SaaS). Grid and Cloud have many commonalities as a distributed computing technology, but each has a unique way of handling problems due to their distinctive service models.

2.2. Cloud Computing

A Cloud [81, 82, 83, 86, 103] is a resource sharing paradigm that allows lots of small real-time resource requests and allocations. It lets us compute remotely on third-party operated and managed computing resources, which include software, hardware and information. There is no provision for queuing allocations until someone else releases resources. This is a different resource allocation paradigm and usage pattern. The allocation and service is driven by economies of scale, and it can be configured and delivered on demand [81, 82, 85, 86, 103]. This approach allows an organization to scale its computing capacities instantly, permitting the organization to accommodate any new requirements without making any changes in its computing infrastructure. It helps to reduce the cost of building and maintaining a scalable infrastructure; they only pay for what they need or use.

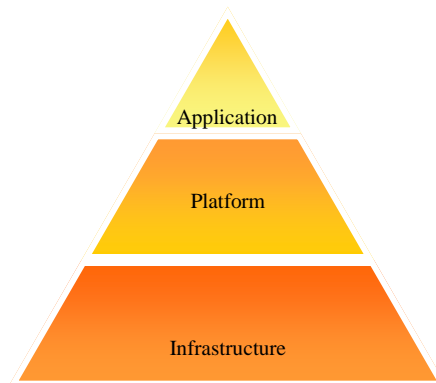


Figure 2-1: Cloud Computing Service Model

Cloud computing providers offer their services according to three fundamental models (see Figure 2-1): Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [81, 82, 83, 85, 86, 103]. IaaS providers supply computers (more often as virtual machines) only. In the PaaS model, cloud providers deliver a

computing platform and/or application stack, such as operating system, DBMS, web server, *etc.* A SaaS provider hosts and maintains the applications in a cloud, and users pay and use the software as a performance guaranteed service; consequently, maintaining their own system is unnecessary.

2.3. Grid vs Cloud

The vision of Cloud computing and Grid computing is the same - to reduce the cost of buying reliable and flexible computing resources. Instead, the resources can be rented from a third-party who owns and operates the resources. Both computing approaches allow multiple users to perform more than one task with several applications at the same time. This approach allows a large pool of users to share high capacity resources and reduce infrastructure costs for each member in the VO.

Cloud and Grid share some problems since they both are fundamentally distributed systems. They both need to manage large amounts of resources and handle large amounts of user requests. They both need to implement often highly parallel and distributed computation. Security is another major concern in both approaches but in different aspects, such as the service models and infrastructures. Cloud computing mainly focuses on the benefits of moving from mainframes to getting on-demand access to low-cost commodity clusters. However, it is operating on a different scale and structure that requires different approaches to address the common problems in a distributed system. Cloud computing faces some unique problems; for example, a Cloud user must determine the schedule and threshold for increasing or decreasing the computing resources bought or leased from the Cloud. This helps to fulfill the demands in the average and peak seasons [87].

Interoperability is another issue that Cloud computing must address. This problem is twofold; first, it inherits the traditional interoperability between distributed systems, like Grids. In order to communicate in such an environment, we need to develop some standards [63, 112, 113] – such as OGSA, WSRF, WSDL, or GFTP. The Cloud community has developed one of the commonly used standards set in mid-2010, OpenStack [112], and the research opportunities between Cloud and Grid have been continuously reviewing in both Grid and Cloud communities. Since Cloud is still in the early stages and is mostly commercially-operated, it is difficult to switch the services between cloud service vendors due to the proprietary APIs and different data formats. This reduces the interoperability of applications and their data between multiple Cloud (computing vendors) [87, 90, 93]. Further, some costs may be hidden from Cloud users. For example, a company could be charged a much higher rate on their network usage because of their storage and database applications, managing terabytes of data in the cloud, or the network latency. These kinds of costs could be avoided by having companies invest in their own infrastructure.

Figure 2-2 provides an overview of the main categories in distributed systems and shows the relationship between Grid, Clouds and other domains [81]. The Web 2.0 concept leads the wave of service-oriented applications and systems and it is one of the core concepts for Clouds Computing. Supercomputing and Cluster Computing mostly focus on non-service types of applications. The foundation of Grid Computing is to share resources for solving resource-intensive problems in a dynamic multiple VOs environment. Grid Computing encapsulates the main features from those few

technologies. However, it is not as application-oriented as Supercomputing or Clusters, nor as service-oriented as Web 2.0 or Clouds.

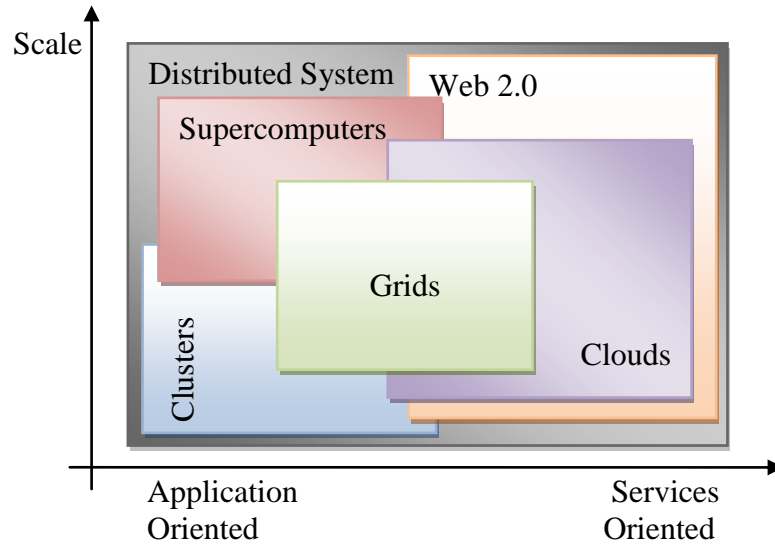


Figure 2-2: Distributed Computing Systems

Grids and Clouds are similar in many aspects, such as functionalities and infrastructure, [81, 86, 103, 113] so they have some common challenges, such as securing a highly distributed and heterogeneous environment [97, 98, 99, 100, 101, 102], standardizing communication protocols, and allowing a high degree of interoperability, [89, 90, 91, 92, 93, 94, 95, 96]. However, they approach the challenges differently due to the difference between their service and delivery models.

Based on the reviewed literature, we believe that Cloud Computing is still at the early development phase; however, many research and development projects are in progress and it is well supported in the community. We also believe that Cloud technology is not replacing Grid Computing [81, 113]. Conversely, we foresee that Grid could be the stepping stone or backbone for Cloud development in many areas, such as authorization, job scheduling, and interoperability. Many lessons learnt from Grid are still applicable to the Cloud environment [113]. For example, most Clouds are currently operated

independently, and their capability to utilize the resources from other Clouds is limited [89, 93]. To make the most of the Cloud potential, many researchers have suggested the use of the “Virtual Organization (VO)” concept (from Grid computing) to improve resource management within Cloud computing [113, 89, 93, 90, 94]. This is exactly the same problem that Grid computing has been dealing with for more than a decade. There is a countless number of research projects in progress, including projects related to communication protocols, authorization models, ontology, and others, which have been done in Grid computing. Likely, the experience from Grid would shorten the learning curve for Cloud computing. Therefore, our research remains focused on the dynamic authorization for Grid environments and will expand to the Cloud environment in the future. To understand the security needs in Grid environments, some existing security technologies and some attacking techniques that are commonly found in a distributed system are reviewed in the next section.

2.4. Security technologies and attacks

To understand the security problems and needs in a Grid, we have reviewed some security technologies and some potential attacks in a real-world environment. The review in this section is not intended to cover all security technologies and problems, but is used to provide more background information for the DRLA development. Two of the most commonly used security technologies and a few frequently found attacks in a distributed system are reviewed in this section. These technologies and attacking techniques may not be directly applicable to the DRLA, but the general concepts of these components definitely help us to visualize the environment and problems for the discussion of the DRLA protocol in the Chapter 3.

I) Transport Layer Security (TLS) / Sockets Layer (SSL) Protocols

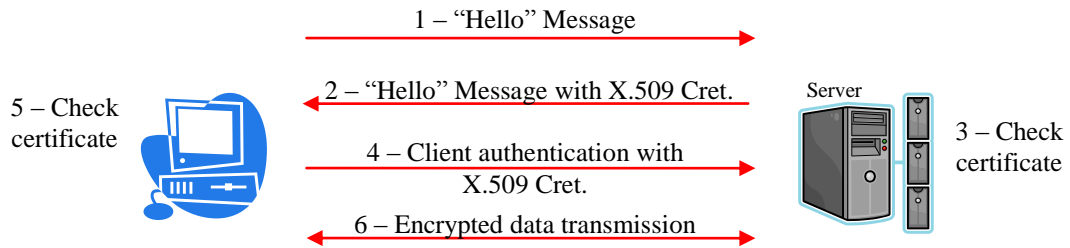


Figure 2-3: TLS and SSL

Transport Layer Security (TLS) and Secure Sockets Layer (SSL) [74] are cryptographic protocols that provide communication security between networks or nodes. The protocols are widely used in many applications on the Internet, such as web applications, instant messaging and voice-over-IP (VoIP). TLS provides end-point authentication by using a digitally signed certificate (e.g. X.509) and increases the content confidentiality by using cryptography with public and private keys in most cases. Figure 2-3 shows a few general steps for using TSL and SSL to increase the security of communication. First, a client sends a request to the server to establish a communication channel. Second, the server replies to the request with a signed certificate. Third, the client should verify that the certificate is signed by a trusted certificate authority (CA). Fourth, the client will send in a client certificate with its private key once the verification is completed. Fifth, the server will also must validate the client certification. Once it has passed the verification, they can use the security key to encrypt the messages.

II) Virtual Private Network (VPN)

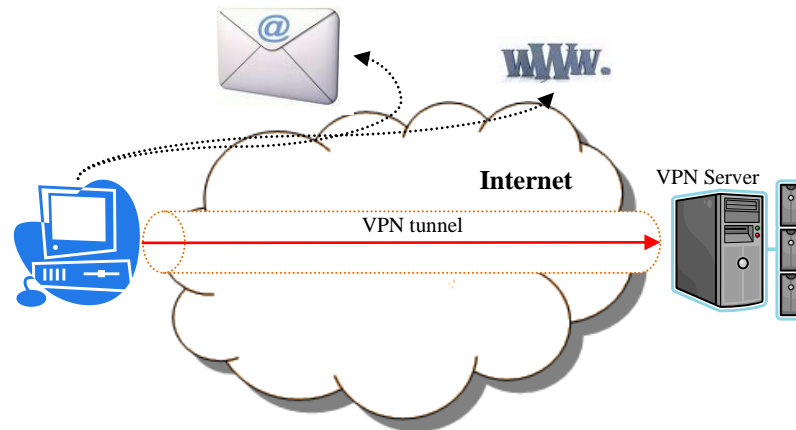


Figure 2-4: Virtual Private Network

A virtual private network (VPN) is a virtual network that is layered on top of a computer network. The data transfer over a VPN is encapsulated in the underlying network. A VPN communication channel can be visualized as an invisible direct link (a tunnel), between the end-nodes in the underlying network (see Figure 2-4). VPN is commonly classified into two types: secure VPN and trusted VPN [4]. A secure VPN provides a mechanism to set up a secure end-to-end communication tunnel for transferring encrypted messages. A secure VPN is often used in a widely accessible network or an insecure network, such as the Internet, to provide authorized users a secure remote access to the private resources. SSL/TLS, VPN, and IPsec are the examples of secure VPN protocols. The main purpose of a trusted VPN is to divide and control the network traffic within a large organization or network; it is not focused on providing security features, such as encryption. ATM and MPLS are examples of the trusted VPN protocols commonly used. The major difference between secure VPNs and trusted VPNs is the level of the security services and the level

of the traffic control in a network. Secure VPNs focus on the security services, such as data encryption. Trusted VPNs concentrate on the data flow control services.

Summarized above are two of the security technologies that are commonly used in the modern computer networks. The review intends to give a general idea of some security techniques and technologies that can be used to secure the communication in a computing system. The rest of this section describes a few attacking techniques frequently found in different networked systems.

III) Denial-of-Service (DoS) Attack

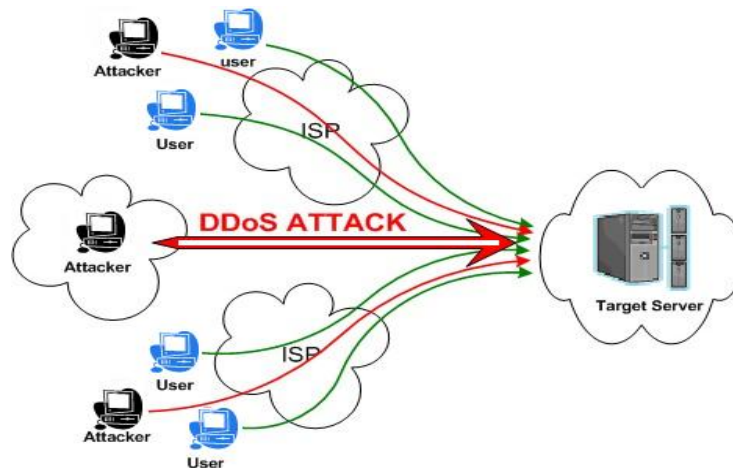


Figure 2-5: Denial-of-Service (DoS)

Denial-of-Service (DoS) attack is characterized by preventing regular users from using the services provided in a system (see Figure 2-5), such as a routing service, e-mail service, domain name system (DNS) service, or web service. Several forms of DoS attack have been used in the past. First, an attacker may send a lot of requests to a victim (server or service) to occupy its resources (such as network bandwidth or a CPU cycle.), so the victim does not have enough resources to handle other requests. Second, an attacker may block the communication path to a victim. This disruption can be achieved in many

different ways. For example, the attacker could induce DNS cache poisoning [75] to disrupt the routing information in a DNS server or service. Another common technique for issuing a DoS attack is to execute a set of operating system (OS) instructions at the victim's machine, causing an OS or machine crash. Each machine or OS that has some weaknesses or bugs can become the target of an attack.

IV) DNS Cache Poisoning Attack

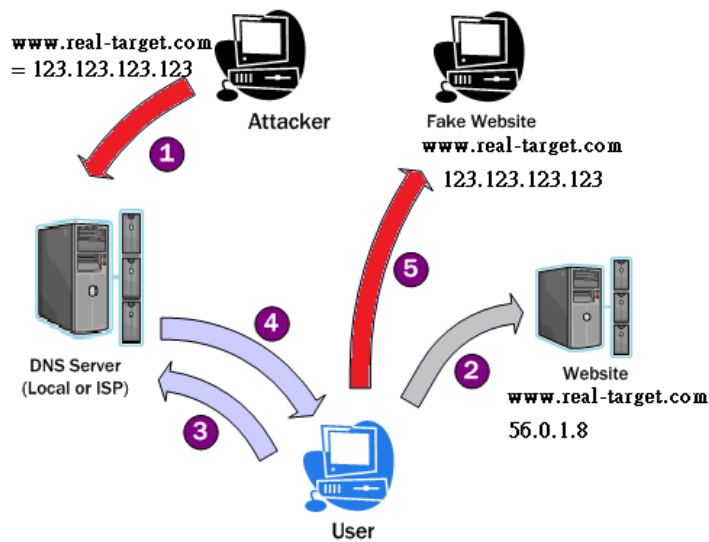


Figure 2-6: DNS Cache Poisoning

A Domain Name System (DNS) translates a domain name to the physical IP address of a resource. The purpose of DNS cache poisoning is to provide un-authorized domain name translation data that is to be cached in a DNS server. Most DNS cache is set up for a performance reason. If a DNS server is poisoned, it may resolve a domain name to an incorrect IP address; and furthermore, it may divert the traffic to another host in the system. If a DNS server does not have a process to validate the source of an incoming DNS response, it may cache and serve the incorrect domain name translation data. Figure 2-6 shows an example of a DNS Cache Poisoning [75] attack. First, the attacker

targets the DNS server used by the user and manages to change (poison) the IP address of 'www.real-target.com' to the IP address (123.123.123.123 in this example) of a web server which contains a fake replica of 'real-target.com'. When the User wants to go to the website 'www.real-target.com' and queries the DNS server for the IP address of 'www.real-target.com', the poisoned DNS server returns the IP address of the fake website to the user. Thus, the user uses the poisoned IP information to visit the faked 'www.real-target.com' at 123.123.123.123, which is controlled by the attacker, rather than the real 'www.real-target.com' at 56.0.1.8. These kinds of attacks are commonly used to direct the network traffic to an attacker's specified location or machine. In most cases, attackers have full control at the destination host, so they can send a Trojan, virus, or spyware to a user's computer, and this malicious content may cause further damage to another user's machine or system.

V) Man-In-The-Middle (MITM) Attack

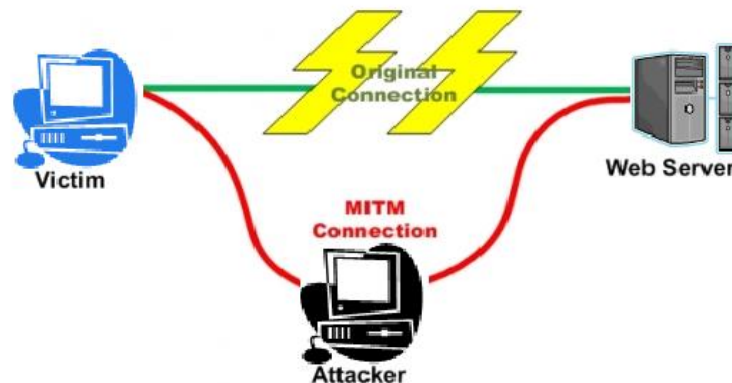


Figure 2-7: Man-In-The-Middle (MITM)

The Man-In-The-Middle (MITM) attack is a form of active tapping in the middle of a communication channel. The attacker makes both end nodes (victims) of the communication channel believe that they are communicating directly with each other

over the connection (see Figure 2-7). In fact, the conversation between these two victims is manipulated by the attacker. The attacker intercepts messages going through the channel and injects some modified or new messages. Many open unencrypted networks often are targeted for a MITM attack. Therefore, many cryptographic protocols have been designed to prevent MITM attacks. However, many of these cryptographic protocols (such as SSL) require users to verify some unsigned digital certificates manually. There are many applications establishing their communications with digital certificates and expect to have a secure channel. However, many users ignore alerts for an unsigned certificate and blindly allow an unsecured communication. This ignorance becomes a big security threat.

VI) SSL MITM Attack

Some research has indicated that some SSL weaknesses are based on weak PKI binding [21, 14, 3]. These weaknesses are a result of a combination of improper implementation, configuration, and usage of the protocol. Burkholder demonstrated a few possible SSL MITM attacks [3].

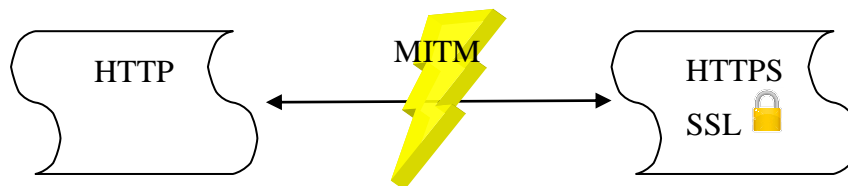


Figure 2-8: SSL MITM attack between HTTP and HTTPS

An example has been summarized in the Figure 2-8. Currently, there are many web applications (such as online banking and online shopping) utilizing SSL to provide a “secure” environment for these online activities. However, a lot of these applications

have not implemented or configured an SSL protocol correctly [3]; the implementation opens some holes for an MITM attack (see Figure 2-9). Many web applications start the login process in an http (non-secure) web page and then transfer it to a https (SSL secured) web page. Since the communication between http and https pages is unsecure, the user's information (including password) can be easily sniffed and masked by an attacker.

At this point, the attacker sits in the middle and proxies the communication between the client and server with SSL. The attacker uses its x509 certificate to communicate with the client and uses the server certificate to talk with the server. Both SSL streams are authenticated and encrypted. This attack does not require the hacker to break the SSL encryption but still allows the attacker to manipulate the communication between the client and server (see Figure 2-9). However, this attack relies on naive users accepting the attacker's certificate despite the warning that they may receive.

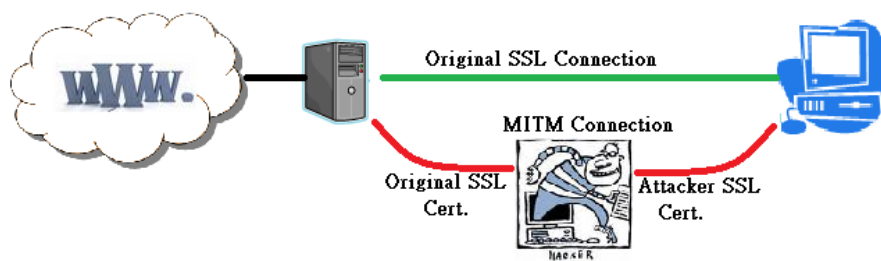


Figure 2-9: MITM attack on SSL certificate

VII) VPN Vulnerability

VPN is a technology that provides a user with an encrypted tunnel to access a firewall protected network and its resources. An organization may set up a firewall to keep intruders away from its network and use VPN to safely encase the communications between a user and the network behind the firewall. However, there are some attacks that

may still be able to jeopardize the network behind the firewall. For example, an intruder may ride through a VPN tunnel, piggybacking on the entrusted user (see Figure 2-10).

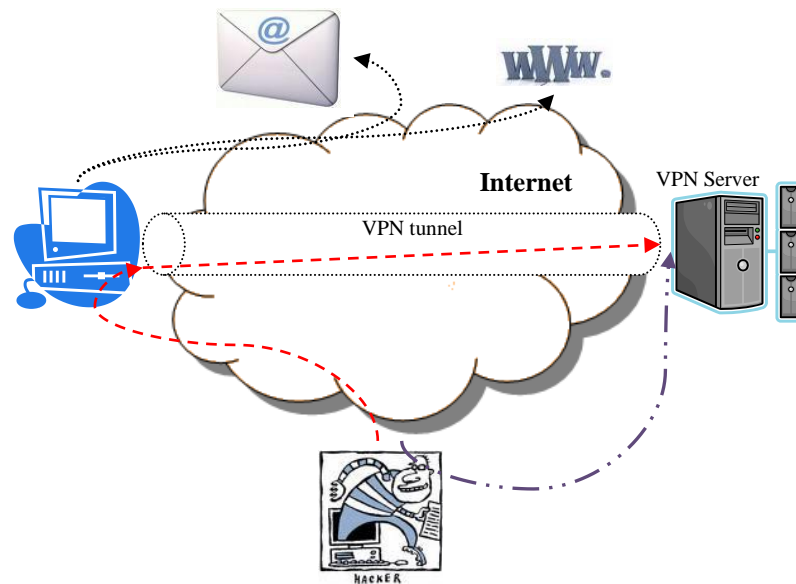


Figure 2-10: VPN Vulnerabilities

In most cases, a VPN client host is not an isolated machine and connected with many other machines, services, and networks, through the Internet. This setting implies that the client host is exposed to some attacker on the Internet. Once the host has been hijacked, the attacker can follow the client through an authorized connection and get right into the network (see Figure 2-10). Another possible attack that can be found in a VPN setting is a denial of service (DoS). In most organizations with a VPN setup, at least one VPN server is configured to manage the VPN connections, and it could become the target of a DoS attack. An attacker may start a DoS attack in the different forms that we discussed earlier.

2.5. Access Control Model [8]

In the last two sections, an overview of a Grid and some general security technologies and problems have been presented. Now, the literature review explores a more specific topic in security, access control, while evaluating some access control models. The review in this section helps us to understand the gaps between the existing models, and hence, helps us to define the DRLA in Chapter 3, beginning with a review of some basic access control principles and techniques. Some dynamic context aware authorization models are then evaluated, and some active and dynamic authorization models are discussed at the end of the section.

1) Principle of Least Privilege

The principle of least privilege [2, 8, 41] suggests that only required permission should be given to a user to execute a task. This principle requires identification of the job scope and the required minimum set of privileges within the corresponding domain only. All other unnecessary rights should be eliminated to avoid any authorized access to the system that are unnecessary. In a dynamic computing environment, many factors may cause security violations within the system, such as bugs in an application, incomplete understanding or misunderstanding of complicated requirements, assumptions made during application development, as well as changes in the environment. Some of these factors may not have caused any problems at the beginning of the life cycle; however, some uncontrollable factors may well introduce a violation later. Therefore, a system should be flexible enough to handle these unexpected security problems, and one of the most commonly applied techniques is to minimize privileges. A privilege is a set of

permissions in a system that is designed to do some tasks that not everyone is allowed to do.

Saltzer and Schroeder discussed security principles specifically and identified minimizing privileges as a principle [41]. In general, there are three basic rules for minimizing privileges in a system [31]. First, granting permission to a program or user who really needs it. The privileges should be only granted to an application or a user who actually needs it and extra permission should not be assigned. Categorizing the users and programs into different roles or groups helps to minimize privileges granted. Second, only give privilege to a process or module that absolutely requires it. For example, dividing an application into smaller sub-programs assists with minimizing the number of privileged modules. Less privileged modules imply intruders have less to exploit. Third, minimizing the active time of a granted permission ensures that there is just enough time to complete the required tasks. Any un-needed privilege should be revoked immediately, rather than being allowed to held while in an idle state. Although, some processes or users may need a privilege often; it is more secure to turn privilege on and off frequently. Once the permission has been turned off, an attacker no longer exploits the permission. In many cases, hackers are only able to trick the privileged application to perform illegal operations while the privileges are active. If the permissions of a program or a user are turned on and off frequently, it is harder for a hacker to hijack a program [31]. This approach may also very easily terminate some unintended processes due to lack of required privileges.

A system which has implemented the principle of least privilege is most likely more stable and secure. If an application were limited to the privileges and time that it really

needs to perform its specified tasks, vulnerabilities in the application could not be used to exploit or crash the system. Thus, the security of the system is increased.

II) Discretionary Access Control (DAC)

Discretionary Access Control (DAC) [42] is an access control method which uses users' identity, such as user name and password, as well as group information to grant access to information in a system. In most systems where the DAC model is implemented, the information owner or administrator is able to transfer ownership or change the rules regarding how an access is to be granted at any time. Therefore, all the permissions and rules are stored on a server and many systems refer to it as an access control list.

III) Mandatory Access Control (MAC)

The Mandatory Access Control (MAC) [42] model labels all information and users with a security level. A security label reflects the relative sensitivity, confidentiality, and protection value of the information and users. A user is only allowed to access the information that is at or below the user's security level, but the user can create information at a level higher than its own security level. In a MAC model, only administrators have authority to change the security label for information and users.

IV) Role Based Access Control (RBAC)

The Role Based Access Control (RBAC) [1] model suggests that all permissions are classified into roles, and users are assigned to the roles based on users' responsibilities, qualifications and permission required. The process of defining roles is usually based on analyzing the organizational goals, structure and security policy. It involves determining who can perform what actions, when, from where, in what order, and in some cases,

under what relational circumstances. Role is the core concept in RBAC model; role hierarchy and constraints are included in some advanced RBAC models [5].

The role concept gives flexibility to the policy designer and system administrator to create and modify security policies, as well as change the mapping between users and roles (permissions). Figure 2-11 illustrates the basic concepts of RBAC (known as the base model, $RBAC_0$) [1]. User, U , represents a human user, autonomous agent, or computer. Role, R , is a symbol of a job function with associated permission and responsibility conferred on a member of the role. Permission, P , corresponds to an approval for an access to one or more objects in the system. The User Assignment (UA) is a many-to-many relation between U and R , $UA \subseteq U \times R$. The Permission Assignment (PA) is a many-to-many relation between R and P , $PA \subseteq P \times R$. Session, S records the mapping between a user and role(s). There are three extended models that are also introduced [1]: $RBAC_1$, $RBAC_2$ and $RBAC_3$. $RBAC_1$ adds the concept of role hierarchies on top of $RBAC_0$. Similarly, $RBAC_2$ adds constraint management onto the user assignment, permission assignment and sessions in the base model. $RBAC_3$ is a hybrid model that combines $RBAC_1$ and $RBAC_2$.

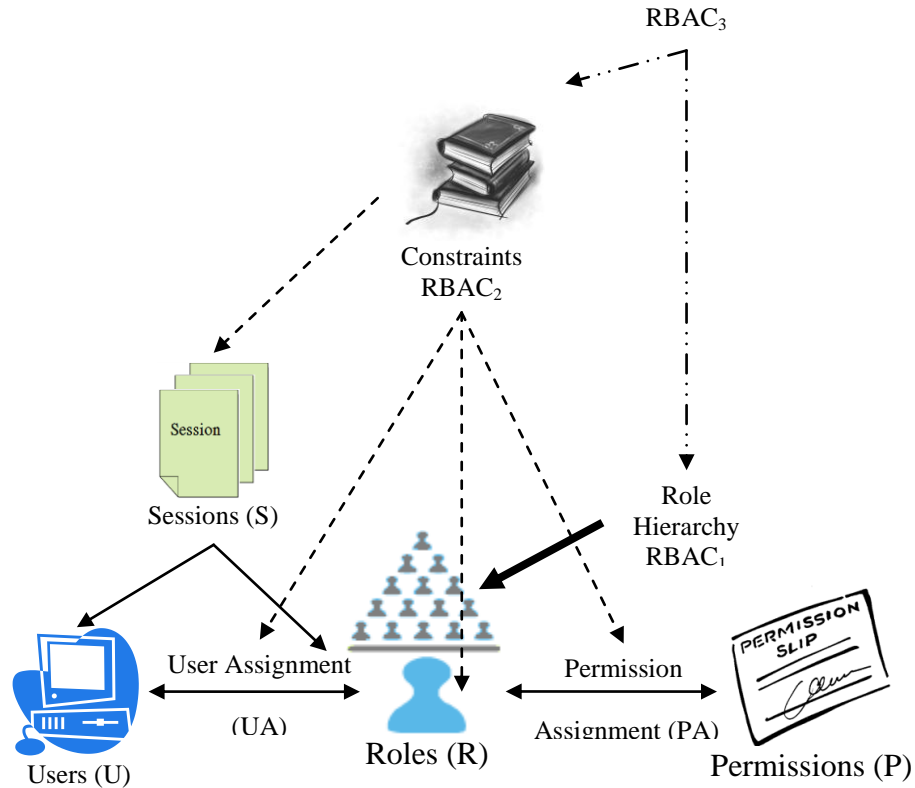


Figure 2-11: Role Based Access Control (RBAC) [1]

V) Dynamic Context Aware Authorization Models

RBAC models have been demonstrated to be an effective and efficient approach for an administrator to manage access in a computing system [1, 2]. Many access control models are extended from the RBAC models with consideration of contextual changes of a user and/or the environment. Several of these models have been reviewed as dynamic context-aware RBAC models and are reviewed in this section.

a) Context-Aware Access Control

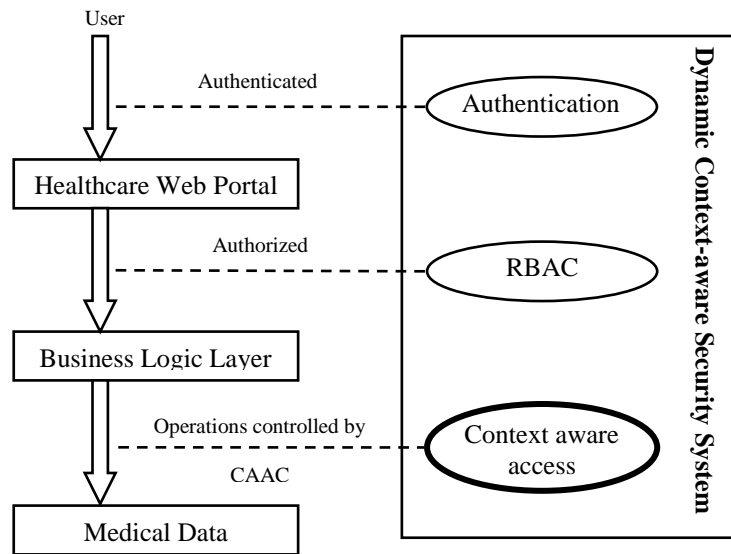


Figure 2-12: Context-Aware Access Control [6]

The Context-Aware Access Control (CAAC) model [6] makes its authorization decisions based upon runtime parameters (contexts) rather than simply the role of the user. CAAC was extended from the traditional RBAC model with an added context awareness module at the end of the authorization process (see Figure 2-12). However, the dynamic facts in the environment and the user's contextual information could be constantly changing during the authorization process. This implies that the contextual information is only considered in the CAAC authorization once. Once a request is authorized and the access is granted, the access remains active (turned on) until the end of the user's session, regardless of any changes to the user's contextual parameters. However, these changes should be reflected in the authorization decision.

b) Dynamic Authorized Role Based Access Control

The Dynamic Authorized Role Based Access Control (D-RBAC) model [9] dynamically grants and adapts permissions to users based on a set of contextual information collected

from the system and user's environments. The D-RBAC model extends traditional RBAC by associating access permissions with context-related constraints. Every constraint is evaluated against the current context of the access request. Thus, the authorization decision is based upon context information in addition to roles. The model can enforce policies automatically because it is not statically bound by an application, and the policy can be modified and applied at run-time.

c) *Dynamic Authorization Framework for Multiple Authorization Types*

Chandramouli develops the Dynamic Authorization Framework for Multiple Authorization Types (DAFMAT) [16] which combines Role-based Access Control (RBAC) and Dynamic Type Enforcement (DTE) with a logic-driven authorization engine. It allows the system to define different authorization types which are used in the authorization engine to accommodate the requirements in healthcare application systems. DAFMAT focuses on the context of an application system, not the operating system, as well as the authorization type for each request. There are three authorization types (emergency, context and non-context) and the corresponding processes have been defined. All of this information is evaluated dynamically. The design goal of adding an authorization type is to reduce the information that is to be processed by the logic-driven authorization engine and hence to reduce the complexity of the logical implications. However, defining some well-represented authorization types and processes that are suitable for most requests could be very difficult. Furthermore, it is extremely complicated to cover all authorization scenarios in a dynamic environment efficiently with only three general authorization types.

d) Extended Usage Control

PL: Pre-defined Local, PM: Pre-defined Multi-domain, DL: Dynamic Local, DM: Dynamic Multi-domain
 PLA: PL Attributes, PMA: PM Attributes, DLA: DL Attributes, DMA: DM Attributes

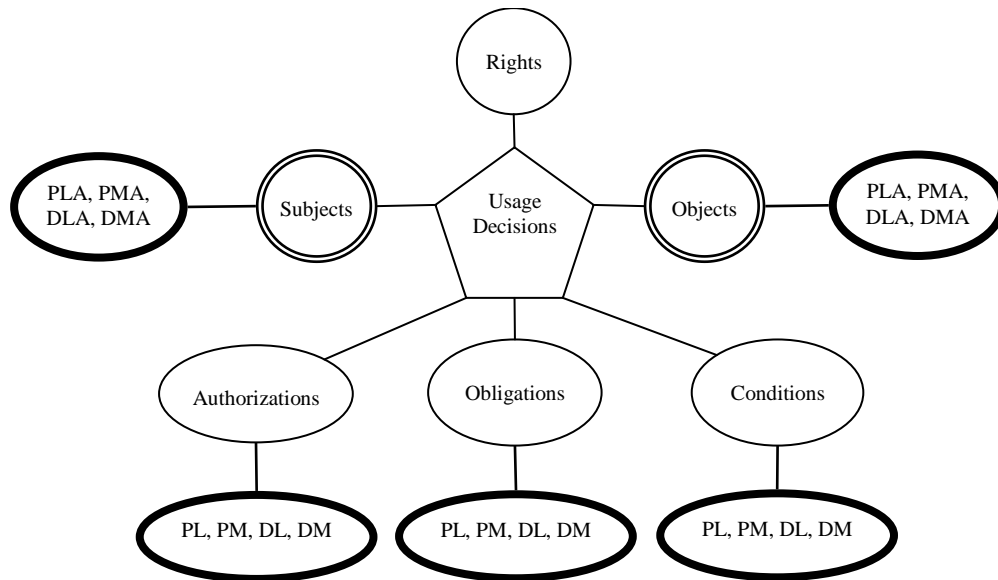


Figure 2-13: Extended Usage Control [17]

Sastry *et al.* [17] re-defined some attributes and components for the authorization process in a multi-domain environment to handle contextual information with the Extended Usage Control (EUCON) model, which is extended from the (Usage Control) UCON model. The extended model tries to avoid some problems in the existing attribute-based access control models (such as UCON). EUCON [17] mostly focuses on how to transfer user information (e.g. privilege, account balance.) from one domain to another. EUCON added many attributes (see Figure 2-13) into the original UCON model to make authorization decisions. Although, EUCON and DRLA share a few common elements, they still have different focuses. For example, both authorization models are designed for a multi-domain environment. However, EUCON handles the authorization problems for users moving away from local domains. The authorization model needs to make sure all the local attributes meet the multi-domain attribute requirements. The problems caused

by the user's contextual changes and resource policy changes in an authorization process have motivated the development of the DRLA protocol.

e) Dynamic Authorization Management Model

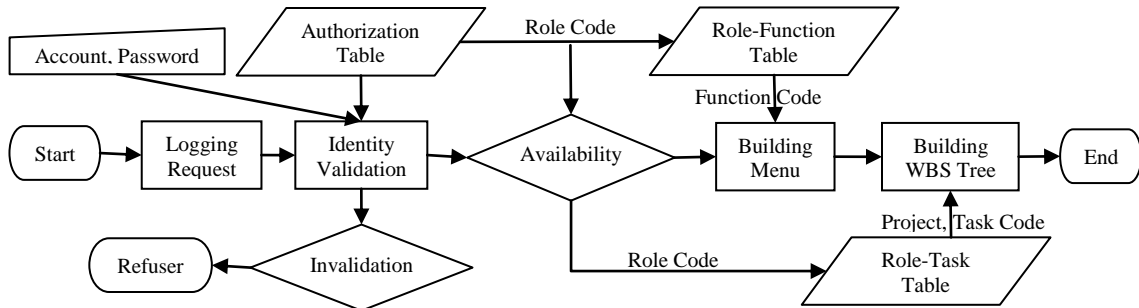


Figure 2-14: Dynamic Authorization Management Model [43]

The Dynamic Authorization Management model [43] based on project management requirements on system function and workflow breakdown. The main idea of this model suggests that authorization should be based on task and role which is a many-to-many assignment relationship between the user and the task. A user should be only authorized for its particular tasks. The authorization process is to assign different function and task items to the user. This assignment process implies that a user may have different permissions in different projects with the same role. Therefore, the context of each request becomes more dynamic. The model is a task-centric model that decomposes a task into smaller sub-tasks, and assigns them to the roles in the system. The dynamic nature of this model is to decompose a task and build the Role-Task relationship dynamically (see Figure 2-14). The dynamic contextual information of a user has not been considered in this authorization model, as it is designed for project management within an organization.

VI) Active/Dynamic Authorization Models

The traditional access control models are suitable for predefined access rules, as they are adjusted manually with policy, environment and user context changes in mind. These manual tasks are time consuming and error-prone. A Grid is an expandable system in which many dynamic factors emerge, and therefore, it becomes impractical to manage access components manually. An active security model provides the abstractions and mechanisms for the active runtime security management of tasks or activities as tasks progress to completion. In a role based active security model, the context of roles or permissions are constantly monitored; hence, a role is activated and deactivated based on its emerging context. RBAC has proven [1, 2, 5, 6, 7, 8, 9, 16, 17, 43] to be cost effective by reducing the complexity involved in authorization management of data. Without active authorization management, permissions will be “turned on” too early or too late in most cases, or remain turned-on long after the completion of a task. Thus, this increases the vulnerabilities of systems. A dynamic authorization approach is needed to grant and revoke access without manual security administration. The following sub-sections, some active authorization models have been summarized and compared to the DRLA proposed in this thesis.

a) Dynamic Role Based Access Control

The Dynamic Role Based Access Control (DRBAC) model [7, 8] uses a Central Authority (CA) that maintains the overall role hierarchy for each domain. When the subject logs into the system, based on one’s credentials and capabilities, a subset of the role hierarchy is assigned to the individual for the session. The CA then sets up and delegates (using Grid Security Infrastructure, 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 status information to dynamically adjust the permissions for each role.

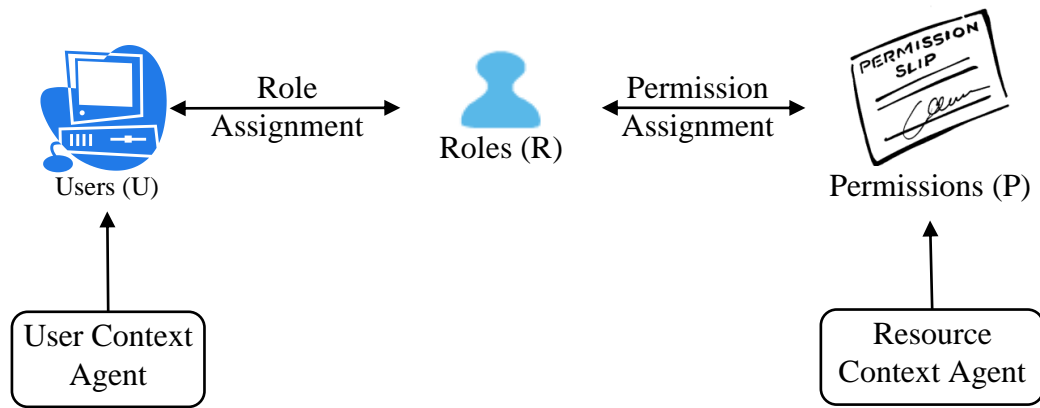


Figure 2-15: Dynamic Role Based Access Control Model [8, 7]

DRBAC handles the dynamic changes in a dynamic environment by updating the Subject-Roles relationship (Role Assignment) and the Roles-Permission relationship (Permission Assignment); hence, it reflects the role hierarchy for the users (see Figure 2-15). However, this approach may not be appropriate all the time. In some cases, the user's contextual parameter changes are not necessarily triggering a change in the Subjects-Roles relationship or the Roles-Permissions relationship. For example, a policy may specify that a user may only be allowed to access the system between 9 am and 5 pm. If a user sends in a request at 6 pm, the user still satisfies the role and permission assignments, thus gaining access under DRBAC scheme. However, the contextual information should be always reflected in the authorization decision.

b) Active Authorization Model for Multi-Domain Cooperation

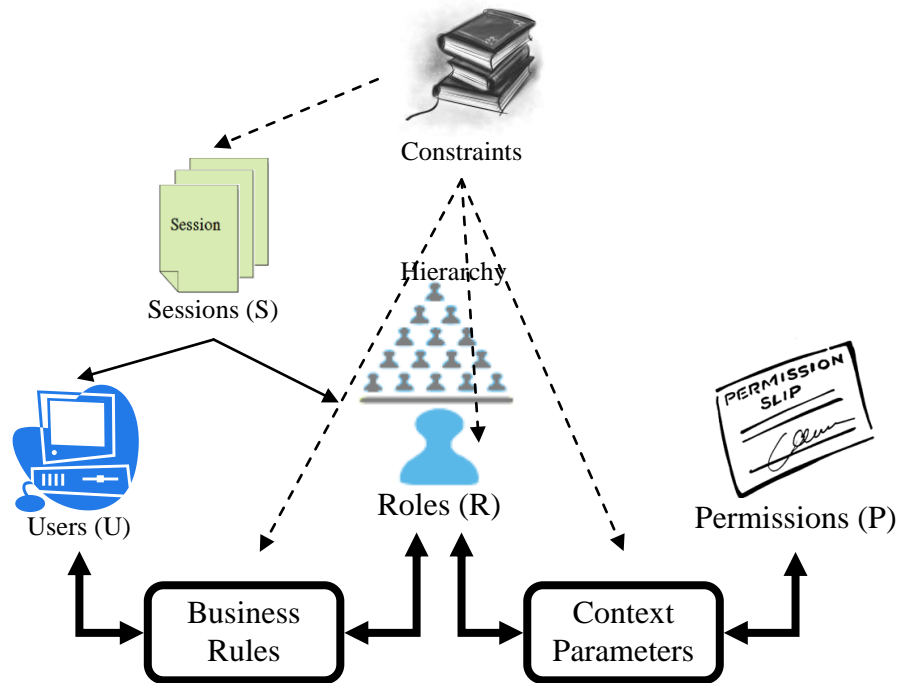


Figure 2-16: Active Authorization Management Model [32]

The Active Authorization Management Model [32] proposes a model for multi-domain cooperation. It updates security policies automatically by applying the concepts of business rules and context parameters to satisfy the dynamic requirements in a multi-domain environment. The model allows policies to be updated dynamically; however, a dynamic policy update may only work on some policy structures that are relatively static unless all possible policy structures can be modeled formally. Any security policy should be carefully designed to fit the organizational needs. Therefore, policy cannot be easily changed unless all business logic and rules can be formally and systemically modeled.

The model has considered the relationship between Roles and Users, as well as Roles and Permissions. Therefore, it introduces business rules (policies) and contextual parameters to represent the relationships, respectively, and they are the keys of the active authorization (see Figure 2-16). However, there are two sets of problems that have not

been addressed in this active authorization model for a “multi-domain” environment. First, the dynamic nature of the contextual information is an important factor for making authorization decisions in such an environment. The authorization model has taken into account the contextual information to make the authorization decision. However, only considering the context during the authorization process is not enough to satisfy the security requirements for a highly heterogeneous and volatile environment. Second, policy management is a very complex area, for example, managing policy conflicts and policy executions both require deep understanding of the business processes in an organization, as well as well-defined and formalized business processes. Moreover, the dynamic nature of a multi-domain environment attaches additional challenges to the issue of policy management.

c) *On-When-Then-Else Event-Based Authorization*

```
RULE [ Rname  
ON Event < Ei>  
WHEN < C1, C2, . . . Cn>  
THEN < A1,A2, . . .An>  
ELSE < AA1,AA2, . . .AAn> ]
```

Figure 2-17: On-When-Then-Else Event-Based Authorization (OWTE) [33]

On-When-Then-Else Event-Based Authorization (OWTE) [33] specifies two categories of events, simple events (file operation, method execution, data manipulation, system clock, and external event) and complex events (composed of one or more simple events). OWTE is an event and RBAC based authorization model that provides active authorization. Figure 2-17 shows the general structure of an OWTE rule. When an event is detected, the authorization rules (On clause) are triggered. Then, all the constraints

(When clause) must then be verified. If a request is authorized, the user should be allowed to execute the requested task (Then clause). Alternatively, the user is able to perform some other actions for a denied request (Else clause). All the OWTE rules are generated from a rule pool and the rules are categorized in three groups: specialized, localized, and globalized. There are two concerns raised in the review. First, the rules in the rule pool have to be pre-defined before the system starts. OWTE may work in a system with a limited or fixed number of events. However, it is very difficult to define and maintain all events in a highly dynamic and heterogeneous environment, such as a Grid. Second, OWTE is an active authorization model, but it has not addressed the problems that may be caused by dynamic changes in a Grid.

d) *Task-Based Access Control*

Task-Based Access Control (TBAC) [36] addresses the anticipated need to automate authorization and related access controls. TBAC adopts the concept of protection states, which represent active permissions that are maintained for each authorization step. The protection state of each authorization step is unique and disjoint from the protection states of other steps. Each authorization-step corresponds to some activities or tasks within the broader context of a workflow. Traditional subject-object models have no notion of access control for processes or tasks. TBAC recognizes the notion of a life-cycle and associated processing steps for authorizations. TBAC dynamically manages permissions as authorizations progress to completion. This differs from subject-object models where the primitive units of access control information contain no context or application logic. The TBAC approach leads to access control models that are self-administering to a great

extent, thereby reducing the overhead typically associated with fine-grained subject-object security administration.

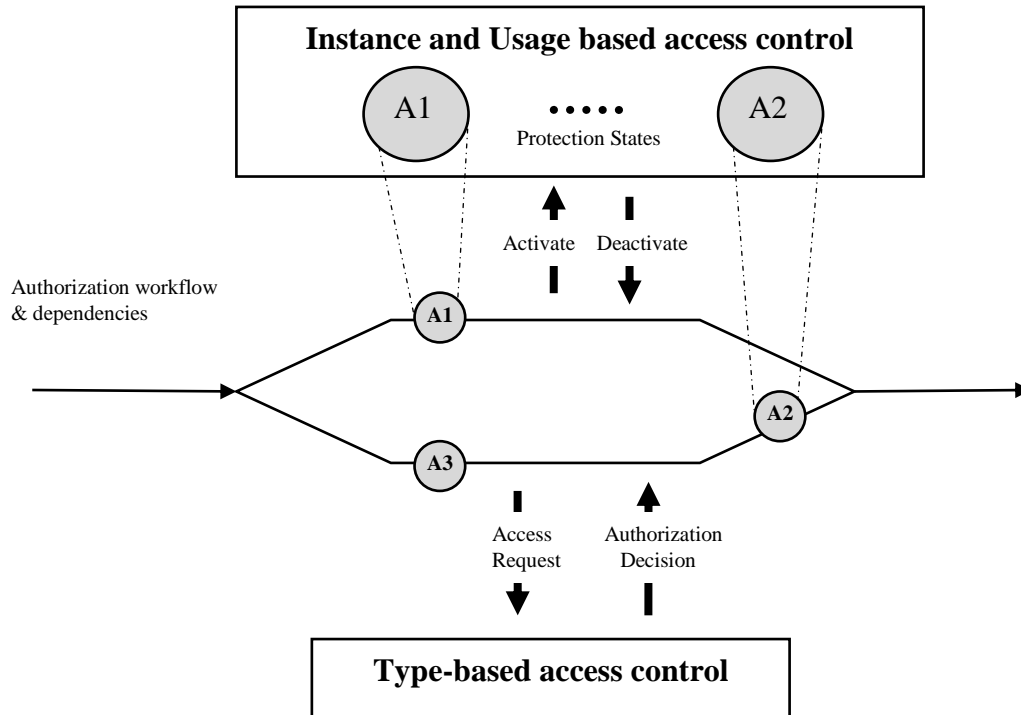


Figure 2-18: Task-Based Access Control (TBAC) [36]

TBAC combines type-based access control and “instance and usage” based access control. In TBAC, a type-based access control module is used to make the authorization decision, and the authorization life-cycle (activation and deactivation) is controlled by the instance and usage based access control module (see Figure 2-18). The focus of TBAC is to manage the flow of the authorization so that there is no specific authorization mechanism proposed in TBAC for making access decisions. Another concern of this approach is that the mechanism for handling any failure in a task is missed in TBAC. This omission is very important in a dynamic environment since many dynamic factors could cause a task to fail or terminate in the middle of a process.

e) *Dynamically Administrated Role-Based Access Control*

Dynamically Administrated Role-Based Access Control (DARBAC) [34, 35] is proposed for access control in web based workflow systems. The concepts of Mission and Objective replace the session or task and context, respectively, to differentiate among different access control needs. Furthermore, these concepts introduced two phases of the access control administration process: access control design (built-time) and access control managing (run-time). During the build-time phase, all the essential components (User, Role, Permission, Mission, and Objective) and static relationships or assignments between the entities at the build time (see Figure 2-19) are retained. The dynamic characteristics in the model are the user-to-role activation, user-to-missions participation and objectives-to-missions binding for a workflow management system. During run-time, users with sufficient administrative permissions can bind an objective instance to a mission instance through an objective-instance to mission-instance binding relationship. In addition, as users participate in mission instances, new entries are added in a many-to-many user to mission-instance participation relationship. In DARBAC, users are only allowed to start a pre-defined mission (task) instance; otherwise, the request will be denied. However, not all missions or tasks in a heterogeneous distributed environment, such as a Grid, can be pre-defined; an exception exists when the scope of a mission is very broad. In that case, the notion of a mission becomes useless. However, the activation, participation, and binding are directly or indirectly defined through the static entities because most of these relationships are defined at build time. Consequently, this model does not address the dynamic user context changes, nor does it have a mechanism to change the assignments dynamically or handle the dynamic contextual information or factors.

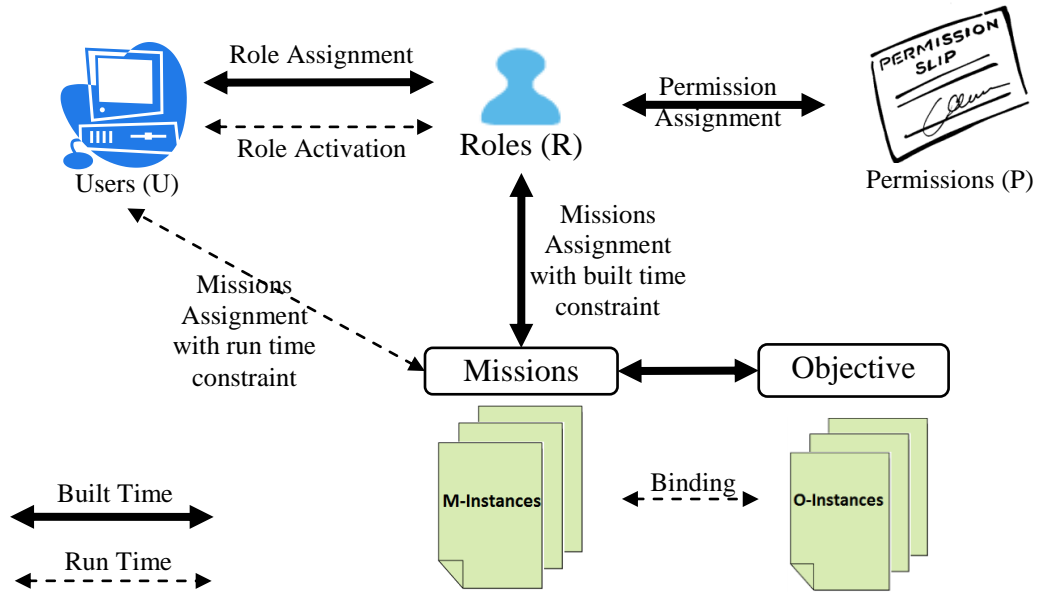


Figure 2-19: Dynamically Administrated Role-Based Access Control [34, 35]

In summary, many reviewed authorization models have considered user contextual information in the authorization process; however, the dynamic changes in the information are not ruminated in the authorization decision processes. To address the dynamic changes is an important security problem that should be handled in Grid environments due to the dynamic nature of a Grid. Many current authorizations are a “turn on/off” process; the permission is granted once the authorized user signs on. This mechanism is not suitable for a dynamic distributed environment, such as a Grid, since the “turned on” permission may be left behind when the user is disconnected from the system due to any unexpected dynamic problem. The “turn on” approach may cause some resources or permissions to be unintentionally held for a long period of time. These problems have motivated this research to develop a new authorization protocol called Dynamic Role Lease Authorization (DRLA) [121]. Before exploring the details of DRLA, the next section will review another core concept in the DRLA protocol, Lease.

2.6. Lease

In this thesis, the concept of leasing is adopted in the proposed Dynamic Role Lease Authorization (DRLA) protocol. Therefore, understanding the leasing concept and any related or similar ideas becomes critical for designing and laying out the detail of the proposed protocol. Applying the leasing concept to different areas in computer science is not a new idea. Much research for adapting leasing or similar time-based techniques (such as time-to-live, ping intervals, and keep-alive [45, 67]) have been conducted since the 1980s. Gray, Cheriton, and Edwards define the characteristics of a lease [45, 46], and outline some criteria and options for managing leases in a fault tolerant distributed system. In their papers, a lease is defined as a contract that gives its holder specific rights over a resource for a limited time period. The advantages of a short or long term lease are also identified. This section provides a review of key leasing concepts, as well as descriptions of how the concepts are applied in some applications. The following paragraphs summarize a number of important characteristics required for understanding the leasing concept.

Leasing is a concept that provides flexibility for a lease grantor and lessee to specify the duration of the leasing period or term, such as limited-term [45], zero-term [45, 48, 49, 50], and Infinite-term [45]. It could be implemented as a contract that gives the lessees some predefined permissions to access some resources (including hardware, software, and information) within the leasing period. Understanding the lease-like (time-based) concepts defined in other research helps to identify the weakness and the strength of the leasing approach. Some file systems, such as MFS [64] and Echo [65], have used the concept of a token to represent the permission of a set of actions (e.g. read, write, and delete) in a file system. By combining the token and timeout concepts, it is possible to

create functionality that is similar to a lease. In a similar manner, DFS [40] uses breakable locks with timeouts, which set a time limit to unlock the token. However, users are not involved in the lock and timeout negotiation process, so users do not get any notification when the status of a lock changes.

Leasing is a way of ensuring a uniform response to any failure that may happen to the associated object [46, 61, 62]. It allows agreements to be made that can then be forgotten without the possibility of unbounded resource consumption, and hence, provides a flexible mechanism for duration-based agreements. Another major characteristic of a lease is its flexibility to renew and negotiate the leasing period and terms. The lease period and terms can either be determined solely by the lease grantor or negotiated between the lessees and the grantor. Thus, every lease has a certain lease time and associated conditions. Before the lease period expires, the client must extend the current terms to continue accessing the required resources. For example, you have to sign a lease when you rent a condominium. The lease provides you the right to use it for a period of time under certain terms. If you need or want to stay longer than the lease period, you must renew and negotiate the lease before it expires. The renewed lease allows you to stay until the end of the new lease period. The owner has the right to refuse the renewal request if both parties fail to agree on the lease terms during the negotiation. In either case, you have to leave the unit before the lease expires.

Selecting the length of a leasing term is a critical process that may affect the system performance. The leasing terms have generalized into two categories in this thesis: short-term and long-term. A leasing term could be simply a time constraint to indicate the expiry of the lease, or it could be more complicated by including some conditions or

actions that the lessee must satisfy. In the next few paragraphs, the major advantages and disadvantages will be summarized and some lease management approaches will also be reviewed. A number of factors must be considered in determining lease length. A lease with a shorter leasing period can minimize the negative impacts caused by failures. For instance, a failure may break the communication path between the resource and user. In many non-lease based systems or systems use infinite lease terms, the resource is locked by one user until the user reconnects to the system and releases the resource. With a short-term lease, the impact is bounded to the end of the lease period so the resource will not be locked forever. In other words, if a lease is not renewed before its expiry, the lease is treated as invalid and the resource will be released. Another advantage of using a short-term lease is that the overhead of managing leases can be reduced, especially in a system that may generate a large amount of leases. There is a trend or expectation indicating that Grid computing has adapted and expanded at a fast pace [52, 53, 54]. Thus, an argument has been made suggesting that a Grid combined with the lease concept is a better method of managing requests. This system allows the possibility of having millions of users and each user making dozens/hundreds of requests concurrently. It implies that there is a need to manage, potentially, billions of leases in a Grid. Therefore, minimizing the cost of lease management becomes an important task in a lease system. Gray and Cheriton developed an analytic model [45] to demonstrate the differences between using a short-term and long-term leases. Short-term leases reduce the resources required for managing leases [45]. The lease management system needs to maintain records for all leases. To manage billions of leases, it could consume a significant amount of computing cycles to handle all the processes. A shorter lease term reduces the management resources that are

required for each lease because the resources that are used for managing a lease can be reclaimed once the lease has expired, and reused in a new lease. Longer-term leases are more efficient for both client and server for the tasks that are performed repeatedly because the overhead of negotiating the leases for repeated jobs is eliminated [45]. Granting a long-term lease means a user or job has the right to use that resource for a longer period of time. In addition, it implies that the granter assuming the security level (or contextual information) of the user remains at the same level throughout the leasing period. Therefore, a long-term lease should only be granted to users who have relatively stable contexts. Another advantage of using long-term leases is that a resource has more flexibility in scheduling jobs because the resource does not need to frequently check the expiry of its leases, and it has the flexibility to reprioritize the tasks, if needed. If a long-term lease has been used, it implies that the system needs to maintain a lease for a longer period of time. We also need to consider the possibility that some resources may be held up by idle processes or leases longer due to failures that may happen in a dynamic environment. Determining the best lease term for a lease-based access control system is a trade-off between minimizing lease renewal overhead and minimizing security risks.

Various approaches have been found in the reviewed literature describing how to manage leases [55, 56, 57]. One common factor for managing leases in these approaches is the relationship between the life cycle of a lease, the application's leasing needs, and the characteristics of the environment. There is no one method that can fit all needs, so one or a combination of the following methods could be applied to address different requirements for a Grid.

1. Each resource agent could host all leases that it issues and provide all utility services to manage the leases, such as renewing and tracking the expiry of the leases. However, this approach may increase the workload of the resources.
2. Dedicate a centralized server to communicate with all resources and users for maintaining all leases. This method addresses concerns about the workload placed on the resource and service providers. However, it does introduce a single point of failure to the system.
3. As discussed earlier, a short-term lease can provide a better fault tolerant mechanism and a long one can provide a more efficient structure for managing leases. In a dynamic environment, such as a Grid, it is necessary to provide fault tolerance in the system or application design. Therefore, in a lease-enabled system, selecting a method to determine the lease term becomes a critical task. There are some techniques applicable to lease management to determine an appropriated length of a lease [56].
 - a. Slow start: The idea is similar to the congestion control technique, a slow start congestion window, in the Transmission Control Protocol (TCP) [68]. We can start with a short-term lease, and then, extend or reduce the term later on. In that case, we only grant a longer lease term to a reliable user after we have collected more statistics regarding the user's contextual information and some frequent occurring events in the system. The decision of increasing or reducing a lease term is based on these statistics. The information gives us a better estimation of the trustworthiness and stability of the users in the dynamic environment.

- b. Event trigger: we can define some events as an indicator of a need to change the length of a lease. For example, if the transfer rates in a computer network are significantly decreased, it could be an indicator of network congestion, and in this situation, shortening the lease term is recommended to avoid a disconnected or unreachable host delaying a resource. Setting up a threshold for the transfer rate is required for the system to detect this event.
- c. Pinging: another way to determine changes is that the system actively enquires about job progress or the contextual information of a user or resource. However, this may increase the network traffic. Therefore, determining the pinging ratio becomes important and should be reviewed periodically.
- d. Trusted task or user: we can safely grant an infinite or long-term lease to some trusted tasks or users. In other words, those tasks or users are trusted to not harm the system or impact other users. This technique also implies that fewer lease renewal requests are required; therefore, the overhead for renewing a lease can be eliminated.

Although previous research has defined various types of leases, it is important to understand how the concept has been implemented in some real systems or applications. It helps us to identify the potential of the leasing approach. Based on the reviewed literature, the leasing concept has been studied and implemented in a few research areas, including file systems, resource management, lifetime management, network management, and memory management. We will now review some applications that currently implement the leasing concept.

I) DHCP Lease "Life Cycle"

The use of dynamic address allocation in DHCP [62] negates the need for a client to own an IP address, but rather leases it. DHCP dynamic addressing enables a machine that communicates with a DHCP server to obtain and maintain a lease of an address [51, 58, 59, 62]. A DHCP IP address lease is similar to a “real world” lease, such as a rental agreement, as it has a similar lease “life cycle”, including allocation, reallocation, normal operation, renewal, rebinding, and release [59]. The lease life cycle, as described in the DHCP standards [62], states that the client moves through the life cycle as it acquires a lease, uses it, and then either renews or ends it. The lease is first assigned to the client through a process of address allocation; if the device is disconnected and the client wishes to re-connect to the system, the system must reallocate the lease. After a period of time, controlled by the renewal timer associated with the lease, the device will attempt to renew its lease with the server that allocated it. If this fails, the rebinding timer will go off and the device will attempt to rebind the lease with any available server. The client may also release its IP address if it no longer needs it.

II) Jini

Jini [46, 61] is a service-oriented architecture and a programming model, which is based on Java technology and used to build a distributed system in a dynamic environment. Leasing is the mechanism adapted for fault-tolerant resource control in Jini in a distributed computing environment. A leasing approach is guaranteed to detect failures and provides a mechanism to automatically clean up the failed components [46]. The essential idea of leasing in Jini is similar to a lease of a rental agreement. When a user asks for access to a resource, the resource can grant access for a period of time that is no

longer than requested, and the lease period is specified in a lease object. A holder of a lease can request to renew a lease, or cancel the lease at any time. If a lease expires, the grantor of the lease can act to reclaim the resource, and lease it to another lessee. A lease includes the detail description of the action that should be taken upon its expiration.

III) Globus Toolkit - Lease-Based Lifetime Management

In many applications or systems, the cleanup process, such as garbage collection in Java, C and C++ [70], are done by custom defined manual operations. If a client dies or the network goes down, it is better to have an automatic mechanism to clean up an unused state. In Grid computing, a mechanism to clean up old, unwanted component states is needed. Globus toolkit adopts the lease concept to manage the life cycle of the resources in the system [63]. Under the lease-based model, resources are removed unless kept alive by interested parties. Any access to a resource will be destroyed when the lifetime lease expires. If any user is interested in using the resource for an extensive time period, the lease must be renewed before the expiry. This is achieved by maintaining "soft state" via WSRP (web service resource properties).

IV) Remoting Lifetime and Leases in .NET

.NET remoting provides a mechanism to communicate between processes in a distributed environment. .NET remoting is another example of implementing a lease based lifetime management. In a dynamic computing environment, clients need to make sure they maintain a lease active on the server. Otherwise, it is possible that a client might be disconnected from the system due to some dynamic failures. A lease represents a period of time that an object will be active in memory before the .NET remoting system reclaims the memory. Each marshal-by-reference (MBR) object has a lifetime that is controlled by

a combination of leases, a lease manager, and sponsors [54, 55, 56]. When a MBR object is called, the .NET Remoting infrastructure creates a proxy object and returns a reference of that proxy to the caller. They are used to manage the leases and the garbage collection processes. The lease manager keeps tracking the expiry time of all leases. If no sponsor renews an expired lease, the lease manager removes the lease and reclaims the memory. The ability to reduce the network traffic by changing the length of the lease terms distinguishes the leasing approach from other methods of managing the lifetime of remote objects.

2.7. Summary

In this chapter, the background information associated with the current state of the related Grid technologies and problems have been reviewed. First, we review some basic characteristics of a Grid in Section 2.1. This research proposes a new authorization protocol that addresses some authorization problems of a Grid; therefore, we reviewed some general security technologies and problems in Section 2.4 that help us to understand the requirements for the new protocol. In Section 2.5, we discussed the access control approaches in the Grid. The last section of this chapter, we review the concept in the DRLA, Leasing. The next chapter introduces the detail of the proposed access control protocol, DRLA.

Chapter 3: DYNAMIC ROLE LEASE AUTHORIZATION PROTOCOL

3.1. Introduction

In this thesis, we investigate one of the key aspects of a Grid – providing secure access to the Grid resources, specifically, the authorization problems of an access control system in a Grid. In this chapter, a new Dynamic Role Lease Authorization (DRLA) protocol [121] is proposed to address those problems in a dynamic distributed computing environment. There are some cohesive relationships among the user’s contextual parameters, system policies, dynamic factors, as well as other elements of a Grid environment; therefore, the proposed protocol attempts to have a better coordination among these factors. The rest of this section describes the motivations and problems that DRLA attempts to address. We will then define the Grid environment that will benefit from this research in Section 3.2 and explore the DRLA protocol in detail in Section 3.3.

Some of the authorization systems are implemented with the “turn on/off” access control approach [36], and we refer to this type of problem caused by this approach as a “long holding permission” problem. DRLA adopts the leasing concept that is discussed in Chapter 2 to address these security problems. The design goal of the proposed authorization is to provide a flexible mechanism, “role leasing”, for dynamic authorization in a distributed computing system, such as a Grid, which is highly heterogeneous and active. In such environments, some resources could be occupied by an idle user (non-active or disconnected from the system) due to some dynamic factors or failures in the system. A role in a dynamic system is a complex construct that includes permission, responsibility, system or organization attribute, *etc.* In this thesis, the focus is

on the permission authorization (access control) in a role based system so we use role and permission interchangeably to simplify the discussion.

The role leasing structure leases out a role to use a resource within the lease period; the permission will be returned back to the system once the lease expires. This approach prevents the permission being held unintentionally for a long period of time. A lease is attached to a permission that is granted to the user, and a leasing period is established. To maintain the permission, the user needs to renew the lease before its expiry. Once the lease has expired, the permission will be revoked, implying that any failure may only hold the resource until the end of the leasing period. For a highly vulnerable environment, a short lease period could be established. For example, a lease period could be set to a minute or even 30 seconds (an arbitrary small number in time). Conversely, a long lease period could be set (e.g. an hour or even a day) in an environment with low vulnerability. The stability and reliability are not only referring to the network environment, but also the consistence of user's contextual parameters.

The effectiveness and efficiency of the DRLA protocol for minimizing the security risk in a dynamic environment are three-fold. First, the leasing structure minimizes the impact caused by a failure in a dynamic environment. Second, it provides a buffer for some dynamic situations that require a short absence or disconnection from the system. For example, a user may require a short disconnection to switch from one access point to another. The user can maintain the access as long as the connection is re-established within the leasing period, and all policies are still satisfied. Third, DRLA maintains the minimum privileges for each user, and dynamically assigns and grants the required privileges to the corresponding authorized role and user. The dynamic contexts and

policies (business and database) are also verified before access is granted. This approach minimizes the security risk that has been identified in the “turn on/off” approach used in many access control systems [36].

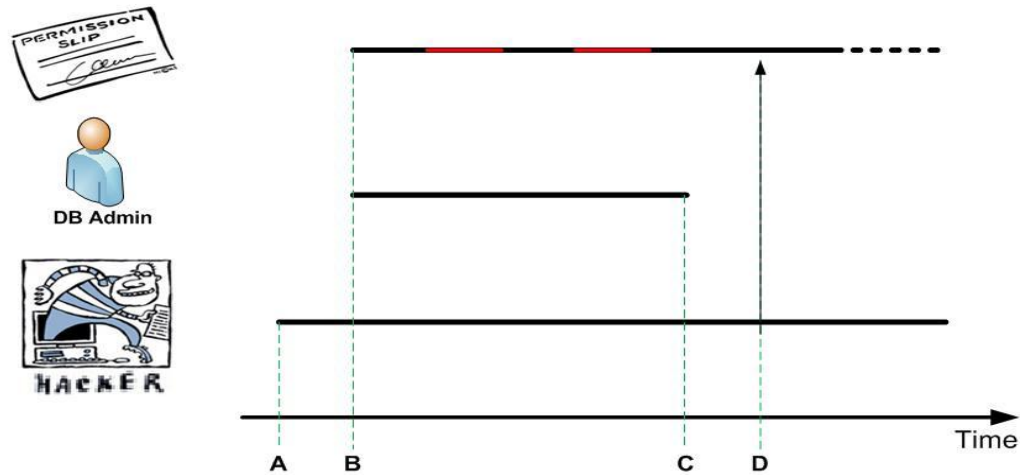


Figure 3-1: Access control system with “on/off” approach

Figure 3-1 and Figure 3-2 are used to illustrate the problem of an access control system with the “turn on/off” approach and provide a high level overview of the “Role Leasing” solution provided by the DRLA protocol (the details of the protocol are described in Section 3.3). Figure 3-1 shows the “turn-on/off” access control method in a Grid. A hacker may tap into the network at time A and monitor all the traffic on the network. He is waiting for a chance to steal information or to issue an attack, such as Man-In-The-Middle (MITM) [3] and Deny-of-Service (DoS) [75]. A user (for example, a database administrator who has the most privileges in the database management systems) logs into the system at time B. The permissions are available to the DBA until his logout (these permissions are called turned-on permission); although, he only needs the permission twice between time B and C (represented by the red line). The DBA becomes disconnected from the system due to a hardware failure at time C. However, the “turned-

on” permissions were left behind since he did not log out properly (these permissions are referred to as left-behind permission). For example, an active or inactive database session could remain in a DBMS after a crash of an application. The hacker may discover the left-behind permission in the traffic trace and issue an attack at time D with some hacking techniques. Many variations of some classical hacking techniques, such as MITM and Trojan, are still able to jeopardize the systems in many cases. For example, a hacker could use Trojan techniques to determine all the session or permission information, as well as exploit the information to initiate an unauthorized action with the MITM method. A few of these technologies and hacking techniques have been summarized in Chapter 2.

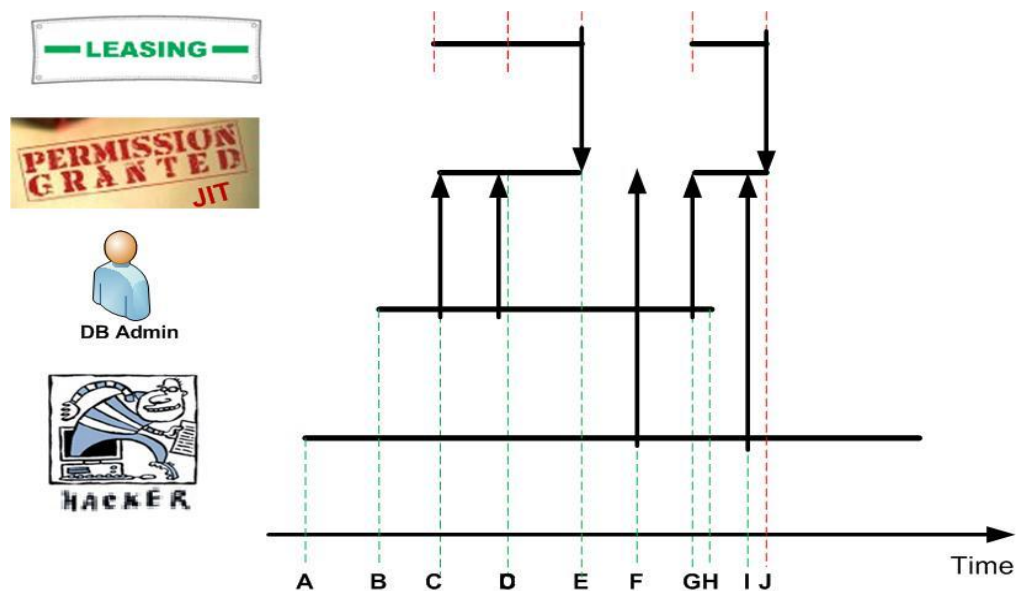


Figure 3-2: DRLA access control system

Figure 3-2 shows a system implemented with the DRLA protocol, which adopts a leasing structure to address the long holding permission problem in a Grid. The DRLA authorization (defined in Section 3.3) takes account of the dynamic factors and grants the required permission dynamically. In this example, the network has continued to be monitored by the hacker since time A and the DBA logged in at time B. However,

permission was not granted at the time of his login. The DBA's first request was sent at time C. Under the DRLA scheme, the required permission was granted and the lease was issued at time C. The lease was renewed right before its expiration, at time D, so the permission was kept alive until the next expiration at time E. The next request was sent at time G. However, the DBA was disconnected from the system due to another system failure at time H. The lease was not renewed before its expiration at time J so the permission was removed at time J to avoid a "turned on" permission being left behind. The hacker issued two attacks in this example. The first attack was fired at time F while DBA was on the system, but, there was no permission assigned to DBA. Therefore, the hacker cannot steal any permission so no damage was done. The second attack was initiated at time I when the hacker discovered that DBA was disconnected from the system and stole the left-behind permission before the lease expiration. The hacker executed a "delete" operation. However, the lease was not renewed and it expired before time J, so the permission was removed. In a highly vulnerable environment, a lease could be assigned at a lower level for each operation or request. The DRLA authorization verifies the lease, contextual information, and other security requirements for each request (the detail of the authorization process is discussed in Section 3.3). In the example above, the hacker is not able to satisfy the authorization requirement and is unable to issue an attack.

In many scenarios, it is quite possible that users do not need all the privileges all the time. As suggested by the Principle of Least Privilege, the permissions should be only granted to an authorized user with no more than what is needed to execute the service requested, and the privileges should be given in the timeframe that they are really needed [30, 31].

The challenge is that Grids are highly heterogeneous environments and have very diverse user groups. Thus, satisfying these criteria is one of the critical requirements to ensure safe and efficient access to a Grid. During the DRLA authorization process, any required and approved privilege will be automatically granted based on the user's context during the execution. The rest of this chapter defines the basic grid environment for the new authorization protocol, and then, reviews the details of the protocol.

3.2. Basic Grid Environment

The proposed protocol, DRLA, was defined on top of a basic Grid environment. This section describes three variances of the basic environment. The first setting is an error-free scenario. Every node knows all components in the system and is fully connected to others. Users are able to use all necessary resources to perform a task without any error or failure. To have a fully connected network, it is required to maintain the global knowledge of the system in each node. However, it is difficult to achieve that in a Grid with hundreds of thousands of nodes. The second case is still an error-free environment, but it is not fully connected to its peers so the communication cost for accessing a resource becomes higher. The last case is a setting which is closer to a real-world Grid. The basic setting is the same as the second case, but error handling is added to deal with the problems caused by dynamic factors and failures in a Grid.

1) Scenario 1: Fully connected error-free environment

The first case is essentially a simplified error-free Grid environment. In this scenario, a few assumptions have been made. First, each self-administrated computer domain or network has at least one node (called a super node, SN) which has sufficient capabilities to handle a high-traffic volume and to provide the required services in the Grid. An SN

also acts as a bridge for communications between its users and the Grid, as well as facilitating communications between SNs to provide core grid features, such as job scheduling and message transfer. All SNs in the Grid connect to each other and form a fully connected network (i.e. a mesh network), as shown in Figure 3-3. In this scenario, the communication pattern is simple; all nodes are connected to each other, so all messages can be sent to the destination in a single hop. Since all nodes are connected and each node records the information of others, such as status changes for peers, the operations in this Grid rely on the global knowledge of the system. Thus, we are assuming global knowledge is maintained and is available to all SNs in the system. It enables each SN to collect the system information and the status of other SNs which facilitates job distribution and related communications. In this error-free environment, all communications are successful in the first attempt. All jobs will be distributed to the first accepting response from a required resource, and any late arriving responses will be discarded without impacting system performance.

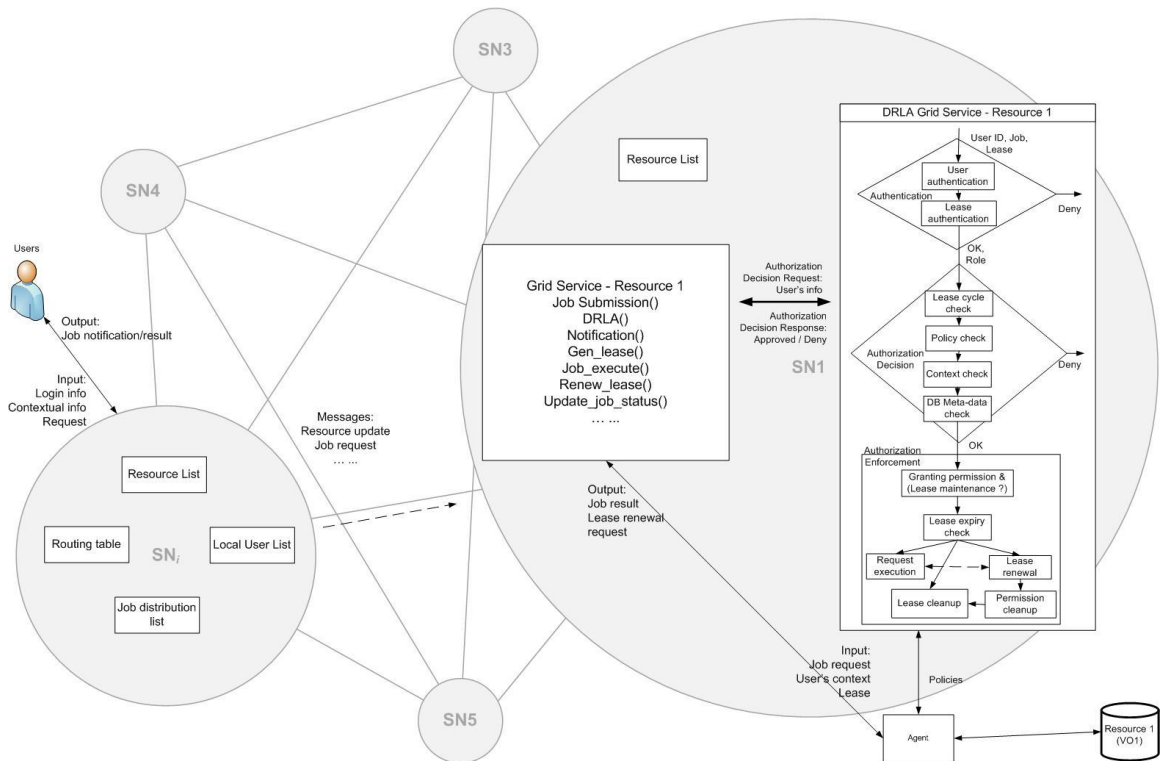


Figure 3-3: Fully connected error free environment

As shown in the Figure 3-3, a grid user connects to a Grid through any affiliated SN in the system. There are a few major components hosted in an SN that are used to facilitate an authorization process with the Dynamic Role Lease Authorization (DRLA) protocol. First, a Resource List can be found in an SN which provides a list of the resources that are available in the system. The list does not identify each individual resource but groups them in categories. For example, all storage drives with greater than one terabyte will be grouped in the “Terabyte Drive” category. Other examples of the groups of resources might be “100-nodes Cluster” or “OCI-128 network”. The second component that is hosted in an SN is a Job Distribution List. The list records all the jobs or requests that are distributed from the SN. Each record in the list contains a few attributes to uniquely identify the distribution of the jobs, including a user ID, job ID, SN ID, and status (accepted/denied). The next element is a routing table, which is used to provide an

efficient message routing mechanism. It records all active neighbor nodes, so the SN can directly deliver a message to another SN in this fully connected system. The last major component is a local user list, which is used to speed up the authentication and the job distribution processes.

The rest of this section describes the detailed steps required for an SN to join the network and to allow a user to access the resources in a Grid. When an SN joins the Grid, it has to execute some initialization processes before accepting a request from users and starting an authorization grid service for any resource it has hosted. Once an SN joins the network, a message will be broadcast to all neighbor nodes to indicate its connection information and the resources it hosts. This information helps the neighbors to update the components discussed above. The neighbors will reply with a similar message containing their information so that the SN can build its routing table and resource list. After these initialization steps are completed, the SN can start grid services for the resources it hosted and wait for the requests on the network.

For a user to exploit a resource in this error-free Grid, there are a few steps required to submit a request through its affiliated SN to the SN of the required resource. First, a user must login to its SN and then obtain the resource list to identify which resources are available and the required resources for its tasks. Once the resources have been identified, the user can submit a request to its SN. On top of the job details, the user's contextual information should be included in the request for the authorization process. The SN sends the request to the SN(s) of the requested resource based on the information from its resource list and routing table. The resource list is an abstraction of resources among different groups, and the user is only able to see the groups (not the individual resources),

so it is quite possible that a request may be sent to all resources in a category. The request will be assigned to the first resource that replies with an acceptance of the task. The acceptance will be timed-out if the resource did not receive an acknowledgement of the response. For each resource, its SN will host two grid services for them. The first grid service is on behalf of the resource agent (called a “resource grid service” from this point onward) to handle a request. Its major tasks include receiving a job, managing the lease of a job, notifying the results of a job, and initializing the authorization process for each request. The second grid service which implements the Dynamic Role Lease Authorization protocol (called the “DRLA grid service” from here onward) provides authorization service for the resource (the DRLA protocol and the grid service will be discussed in Section 3.3).

Moving away from a tightly coupled Grid described above, most Grids do not have a fully connected network topology so not all nodes are directly connected to others. In most cases, a communication channel is constructed through a multi-hop path. In a Grid that is composed of hundreds or thousands of nodes, it is almost impossible to maintain the global knowledge of the system at each node. The second scenario changes the communication mechanism to compensate for these challenges.

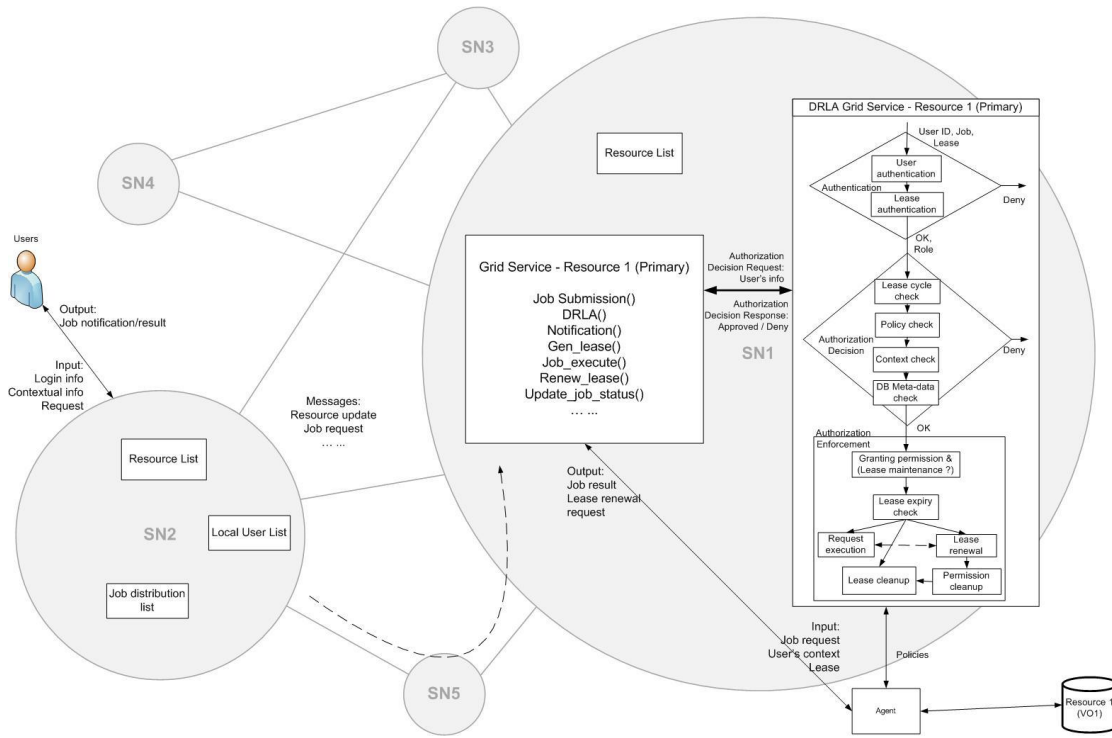


Figure 3-4: Partially connected error free environment

II) Scenario 2: Partially connected error-free environment

The second setting is an extension of the first scenario. It retains an error-free environment. However, not all nodes in the system are fully connected (a non-mesh network), and global knowledge of the system is not available. This setting implies that most communications between nodes rely on multi-hop communication paths (see Figure 3-4). In such multi-hop communication environments, many challenges, such as routing algorithms, security, and message management, have been studied in the last decade. Another challenge in this scenario is the absence of global system knowledge. These gaps in knowledge increase the difficulty of identifying an efficient routing path. A Peer-to-Peer (P2P) [29, 76, 77] network architecture is one of the most popular and well studied networking structures in the area of multi-hops communication. However, it is difficult to determine an efficient routing map without the global knowledge in this scenario. Thus,

we have adapted the broadcasting communication concept in the Grid for this thesis to illustrate and prove the concept. This method allows the nodes to communicate with others without the global knowledge and/or the need to maintain it. Another advantage provided by the P2P structure is that the routing performance could be increased as the size of the network increases and more routing paths become available in the system.

In summary, two of the most commonly used communication methods are broadcasting and selective broadcasting [71]. The broadcasting approach is the simplest method to implement. A message is broadcast from one node to its neighbors and all neighbors take the same approach to forward the message until the message arrives at the target node. This communication method may generate large amounts of forwarding messages and fill the system's communication channels. Selective broadcasting algorithms (e.g. DHT [72]) could be used to reduce the amount of broadcast messages, thereby providing a more efficient routing method in terms of the amount of messages. Since there is no direct link to each node in the system and no need to know the information of all nodes, the way to manage the job distribution list should be changed in this situation.

Job distribution list in SNI

Scenario 1:			Scenario 2:	
Job ID	SN ID	Access	Job ID	SN ID
1	1	1 st Granted	1	1
1	3		2	
1	4	Granted
1	5	Granted		
2	1	Deny		
2	3	Deny		
2	4	1 st Granted		
2	5	Granted		
...	...			

Figure 3-5: Job distribution lists

In scenario one, we have an entry for each direct link connection which indicates the access decision (see Figure 3-5). In the second scenario, we only record one entry for each job and the destination will be filled when the first acceptance of the corresponding request is received.

In summary, a conceptual error-free environment in a Grid has been outlined above. An error-free network environment is either fully or partially connected, and guarantees each node is reachable from others in a direct or indirect (multi-hop) communication path. Another important aspect discussed in an error-free environment is the availability of the global knowledge of the system. The global knowledge could be very helpful to establish an efficient communication channel in a Grid environment. Based on this conceptual error-free Grid environment, we have made the following assumptions. First, no failure happens in the system. In other words, all physical network connections are connected all the time, and no message will be lost in any communication. Second, we assume that the existing network protocol (such as Open System Interconnection - OSI model) [73] provides a reliable network environment for message exchange and a flexible network structure, so a system can be expanded as needed.

III) Scenario 3: Error handling implemented environment

In this section, we discuss some common problems that could happen in a Grid environment. This is the main difference between the third scenario and the ones discussed above, namely, Grids with error-handling. In a dynamic environment, many factors could affect a job's execution cycle. For example, resource contention or network congestion may increase the execution time or cause a user or resource to become

unreachable. The method that has been adopted in this thesis to handle these problems is a timeout based structure called leasing [46]. If any job or message is not returned before the lease expiration, the job will be determined to be an incomplete task, the job or request will be released back to the system, the job could be allocated to other available resources. This method eliminates any incomplete jobs with an expired lease holding up the resources, or prevents any disconnected resource, which is unable to continue providing service. One of the major advantages of the leasing structure is that it effectively reducing the impact of the problems described above by employing the timeout solution. The leasing method also provides a flexible mechanism allowing a resource to renew a lease for a job based on the dynamic parameters during the execution. Thus, a progressing job will not be determined to be an incomplete task and will be released back to the system. This method allows a resource to handle a long-running job in a flexible way, as well as providing a status update for a job by tracing the lease status.

Another commonly raised concern of the dynamic factors in a Grid is the security issue. The proposed protocol handles one of the major security issues, the authorization process; the protocol combines the leasing structure described above and a new dynamic authorization. It maintains the minimum privileges for each user, and dynamically assigns and grants the required privilege to the corresponding authorized user. The dynamic context (business and database) policies are also verified before the access is granted. This approach satisfies the Principle of Least Privilege, and hence, minimizes the security risk identified in the “turn on/off” approach [36]. Satisfying those two aspects and handling the dynamic nature are critical in a Grid access control system.

3.3. Dynamic Role Lease Authorization (DRLA) Protocol

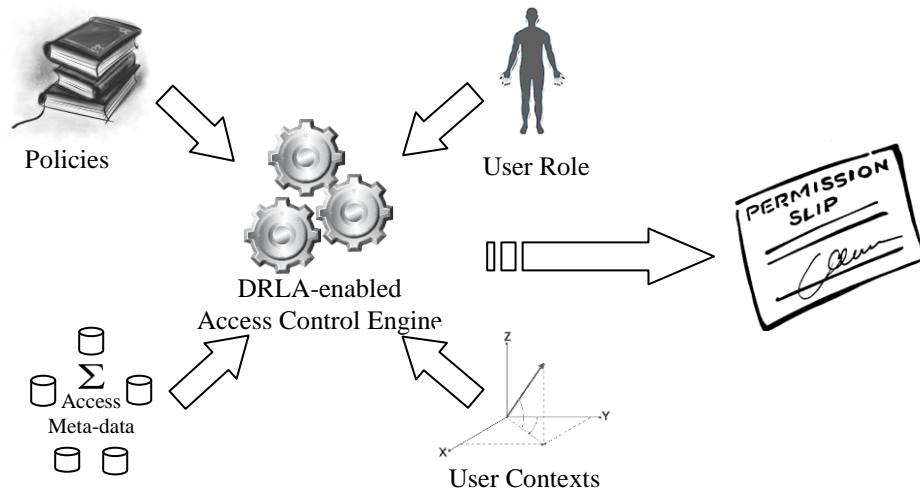
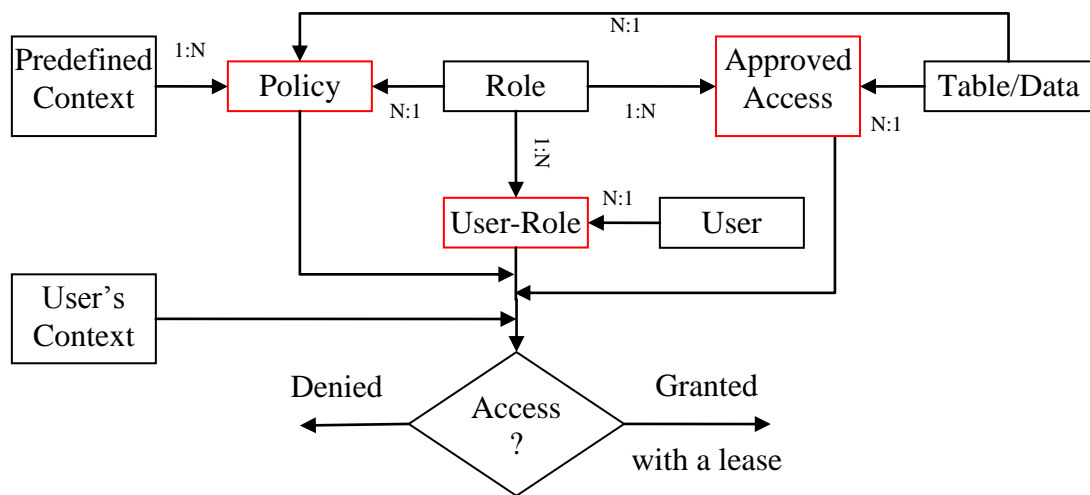


Figure 3-6: Conceptual Diagram of DRLA

Any user in a DRLA-enabled Grid (a Grid implemented DRLA protocol) can send a job request to a resource in the system through its scheduling service. The access request and user's contexts are then forwarded to the DRLA-enabled access control engine for the corresponding resource, which is implemented as a Grid service, to authorize the required access. Once the access is granted, the user can use the requested resource to perform the tasks or obtain related information. Figure 3-6 is a conceptual diagram of a DRLA authorization service. The DRLA-enabled engine is based on the user's role, the user's context (such as time and location), the meta-data of the users' privileges, and the resource policies to authorize the user and grant the required access.

To implement the DRLA protocol, each resource has to provide the information about Users, Roles, Predefined Context, and Table or Data in the resource that defined or registered in its system for its DRLA grid service to make the authorization decision. Figure 3-7 shows the relationship between the components or objects of the DRLA protocol.



Dynamically update and refresh is required for any dimension change

Figure 3-7: Entity Relationship Diagram of DRLA Components

The following tables show an example of the metadata entities Figure 3-7. The title labeled “Role” is a list of existing roles in this system, and “Predefined Context” is a list of criteria that may be verified in the authorization process. “User” is a list of users in the system. Each record contains information about the user.

ID	Name	Store Num	Yr of Service
321456	Bob	101	1
159753	Tom	201	15
789321	Zoe	301	6

User: a list of users in the system

Context Name
Time
Subnet
Location

Context: a list of criteria that a policy may be used in the system

Role Name
Administrator
Manager
End User

Role: a list of roles in the system

Table Name
Sales_Fact
Product_Dim
Cost_Fact

Table/Data: a list of tables/data available in the system

The Policies, User-Role assignment, and Approved Access represent the relationships between those components. These components could be implemented as a set of tables in a relational database, a set of objects in an object-oriented system, or in another form. To simplify the illustration, we assume that all these components could be predefined by experts in a relational database system at the initial stage and be maintained by the grid service that we define later in this section.

A Policy object defines the relationship between the predefined Context and Role objects, while specifying the type of access, targets and conditions. The table below shows an example of some policies set up in the system. For instance, policy #1 specifies that a user with the End User role could access all tables between 9am and 5pm only. The dependency between policies can also be defined as one criterion, as well. In a distributed dynamic environment, a granted access (specified in Approved Access object) may be subject to certain conditions, such as time, location, and other dynamic factors in a Grid so the policies could be changed frequently to reflect the security needs. Therefore, a Policy object has been defined in DRLA to correlate the conditions to the corresponding role and resource, and it also facilitates the dynamic policy verification in DRLA. Policy management (such as policy creation, conflict resolution, *etc.*) is a different research topic, so it is not included in the scope of this thesis.

Policy ID	Context	Role	Min Value	Max Value	Depended Policy	Table Req.	Action
1	Time	End User	9	17			Allowed
2	Subnet	Manager	196.128.1.0	196.128.2.0	1		Allowed
3	Location	End User	PEI	PEI			Denied
4	Location	End User			1		Allowed

Policy: a list shows the access control policies of the roles based on the contexts

The Approved Access object records the privileges that have been approved but not necessarily granted for each role. Each privilege should only be granted when it is needed. For example, the following table shows a simple approved access relationship between Table and Role in which roles are allowed to access a particular table and the privilege status.

Table	Role	Status
Cost_Fact	End User	Approved
Product_Dim	End User	Granted
Product_Dim	New User	Approved
Sales_Fact	Manager	Granted

Approved Access: a list of approved access of each role

In a Grid, the relationship between Role and Table or Data could change frequently. For example, much new data could be generated in a system and the new data is only accessible to certain roles or users. Therefore, the Approved Access object will be updated to reflect the access control on the newly generated data. Similarly, some old-data may be deleted from the system, so some users in a particular role may lose the corresponding access privilege to the data. Therefore, the Approved Access must be modified to eliminate any out-dated records. In summary, the Approved Access is used to maintain the dynamic privilege status.

Role	User
New User	Anyone
End User	Zoe
Manger	Tom
Manger	Bob
Administrator	Bob

User-Role: a list shows the assignments of users into roles

In the same manner, a User-Role object represents a relationship or assignment between User and Role. Each user should be assigned with at least one role, and each role could have multiple users. For example, the table above shows a simple User and Role assignment; Zoe is assigned with the End User role, and Bob is assigned with both Manager and Administrator roles. User-Role assignment could be changed very often; therefore, the User-Role assignment will be changed all the time to reflect the changes. Another dynamically changing element is the user's context. In a dynamic distributed environment, for example, user can move from one location or organization to another, or even switch roles within an organization, and the change may results in the re-assignment of the role. The User-Role is used to track these changes. To illustrate the application of the DRLA protocol, the next two sections will discuss the detailed structure of the protocol and then provide a use case to illustrate the authorization process.

l) Physical structure

Before we get into the core of the DRLA protocol, we must describe the surrounding environment or components that connect to the DRLA grid service. Figure 3-8 shows a typical example of a Grid implemented with the DRLA protocol. Each resource has two grid services: resource grid service and DRLA grid service. The resource grid service handles the job submission and manages the leases on behalf of the corresponding resource, and the DRLA grid service is responsible for the authorization process.

Multiple instances of these grid service pairs could be implemented in the system to increase the availability and reliability of the services. The DRLA protocol specifies the input, output and implementation detail of the DRLA grid service, and the other non-authorization supporting services could be implemented accordingly by each system that adopted DRLA.

The following describes general steps of a Grid implemented DRLA protocol handles a job request as illustrated in Figure 3-8.

- 1) A user logs into the system by sending an ID and password to the associated super node, SN2 in this example.
- 2) The user obtains the resource list from its SN, which recorded all available resources in the system.
- 3) The user may send a request or job, which specified the required resource and tasks to the SN. In this example, the user would request Resource 1 to perform a specific task.
- 4) An SN forwards any requests for a resource that is hosted by another SN. In this example, SN2 received a request for Resource 1, so it forwards the request to its neighbor SNs, as SN1 is not the primary host of the resource grid service of Resource 1. Each neighbor SN takes the same approach to forward the requests. With this approach, the request could reach the primary host directly or indirectly through one or multiple SNs.

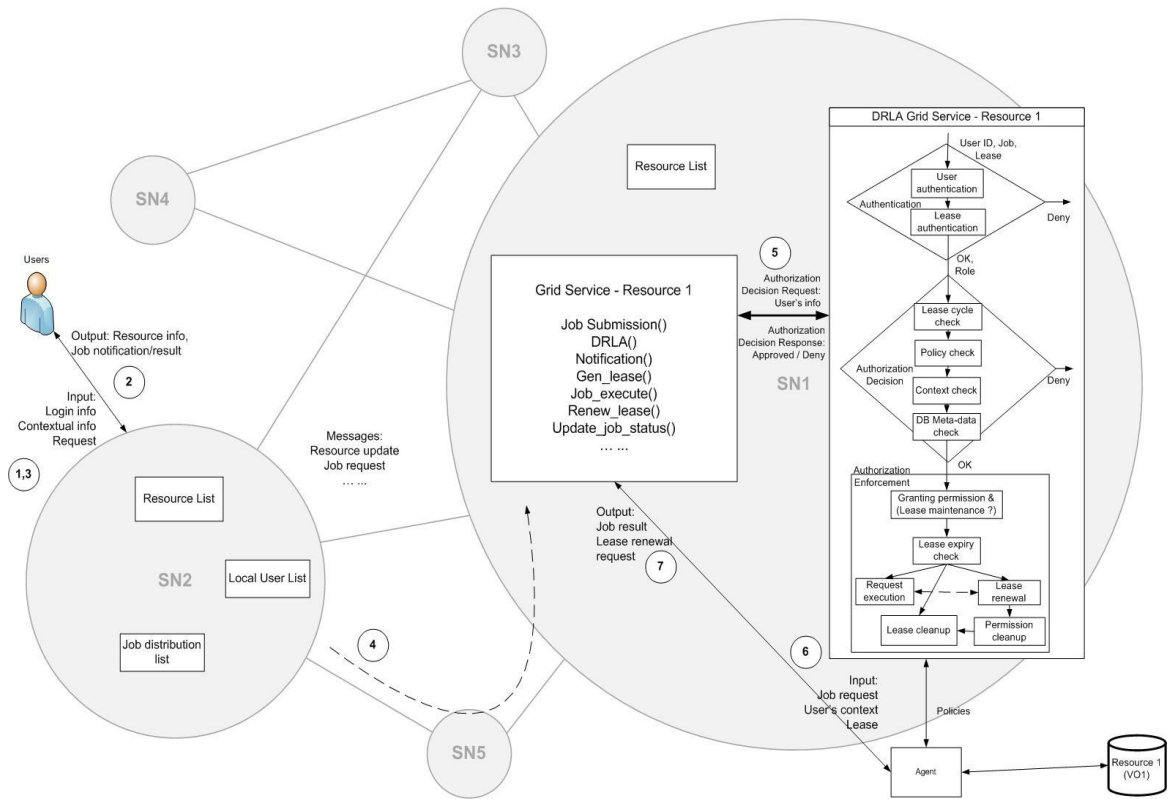


Figure 3-8: DRLA-enabled Grid

Once the resource grid service receives the request, it verifies that all required information (such as ID, user's context and job description) is attached to the request. Before sending the request to the corresponding DRLA grid service for Resource 1, the resource grid service generates and attaches an unsigned lease to the request. As defined in Section 2.5, a lease is a contract that gives its holder specific rights over a resource for a limited time period. An unsigned lease represents a conditional contract subject to the approval from the DRLA grid service. A basic lease could be a pure time based contract to indicate the expiry time of the contract, and it may provide some basic functions to maintain the lease operations, such as a renewal, lease verification, update, and removal.

```

define struct Lease {
Lease_ID: str(10);           // Lease ID
Issues_Date: Date;         // Date time of the lease issued
Lease_Term: Number;        // Lease duration
Renewable: Boolean;        // Lease can be renew or not
Leased_Permission[10]     // List of permissions associated with this lease
Other_Info: str(100);      // Other Information
... ..
getID();                   // Return the ID
renew();                   // Extend the expiry date
Boolean check_Lease();    // Return True if the Lease is valid, otherwise, return False
update();                 // Update the contract information
remove();                 // Destroy the lease
... ..
}

```

In the basic data structure above, it defines some basic attributes and functions that should be found in a Lease object. The basic attributes may include *Lease_ID*, *Issues_Date*, *Lease_Term*, *Renewable*, or *Leased_Permission*. A *Lease_ID* in a Lease object is a unique identifier in the system which is used to look up a Lease. Since a Grid could be a federation of different administrative groups or domains of users and resources, setting up one commonly used sequence number for Lease-ID is very difficult. Therefore, the following format of a unique Lease-ID is used:

Resource IP address / type / local sequence number

An IP address of a resource is a unique identifier for a host in the most modern networked computer systems. The type in the ID represents the type of the object, *L (Lease)* in this case. A local generated sequence number is the third component of the ID. The sequence number is generated by the resource, so it is not required to get any consent from other neighbors in the Grid; hence, it simplifies the formation of a unique ID. The combination of these three components provides a unique ID across the Grid.

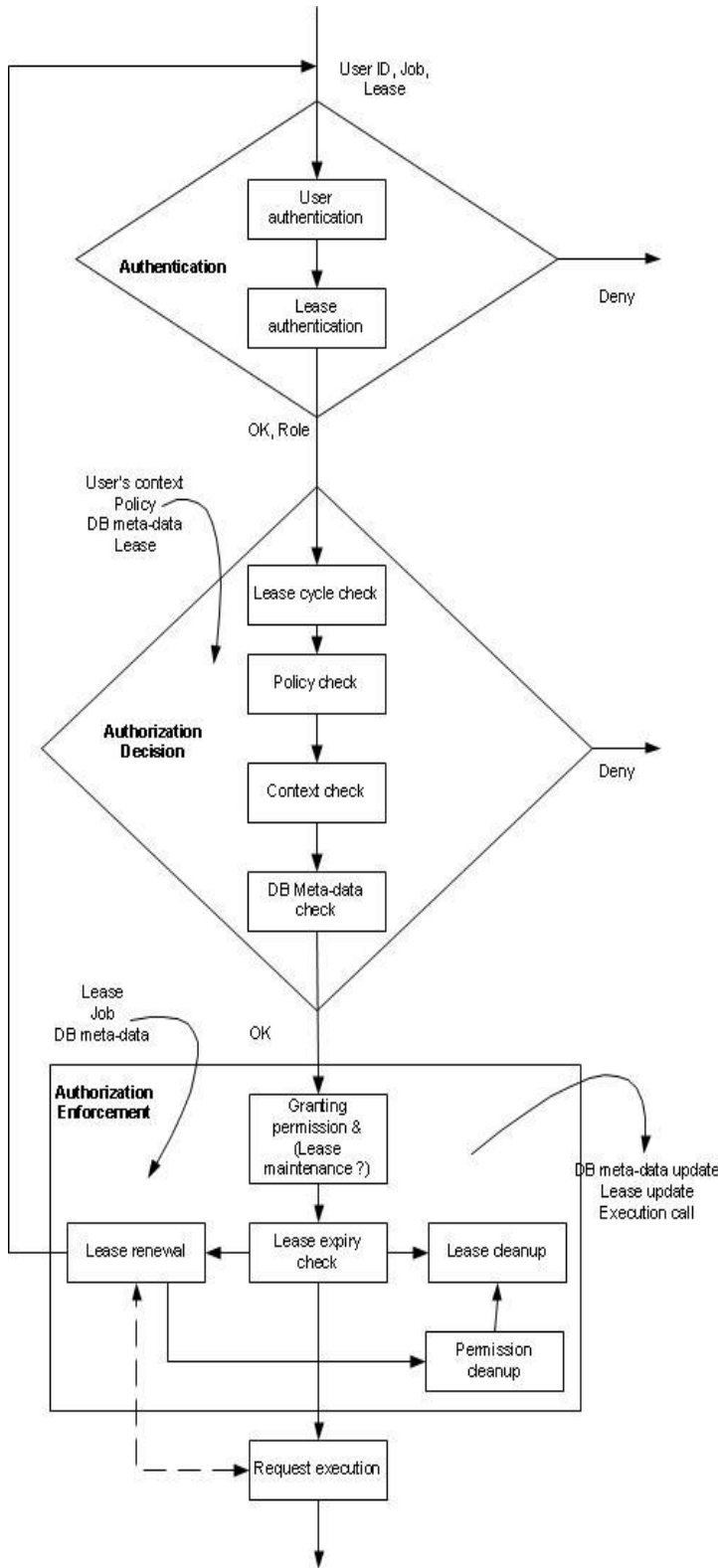
Issues_Date records the date and time the lease was created, and the *Lease_Term* indicates the lease period of the lease. Both *Issues_Date* and *Lease_Term* are used in the function, *check_Lease()*, to determine its expiry. Before a lease expired, the lease can be renewed by calling the function, *renew()*, if and only if the Renewable flag has been set. The lease data structure described here is just an example, and it can be modified to fit different requirements in each system.

- 5) The DRLA grid service forwards the user's ID, lease and contextual information as the initial input to the corresponding DRLA grid service to make the authorization decision. There are three components in a DRLA grid service: Authentication, Authorization, and Authorization enforcement.
- 6) The job will be sent to the resource local scheduler once an approval has been returned from the DRLA grid service. The agent is also responsible for maintaining and renewing a valid lease status.
- 7) Once a job is completed, the job result will be returned to the resource grid service. A notification is then sent to the user and the status of the job is updated so other components, such as the DRLA grid service, will receive a notification.

II) DRLA Authorization Process

This section describes the details of the DRLA grid service. There are three main groups of procedures: authentication, authorization decision, and authorization enforcement (see Figure 3-9). The authentication (step 1) in the DRLA grid service ensures the job submitter is a valid user of the resource and the job is submitted with a valid lease. The user authentication (step 2) is required because a valid Grid user does not imply that the user has a right to access all resources in the Grid. The second important task (step 3) in

the authentication is to validate the lease associated with a job; this task ensures the lease is still valid in the SN which issued it. The main purpose of the authorization steps (step 4) is to make an authorization decision about granting the requested access for a job based on the dynamic factors in the system. This involves verifying the lease term set in a job (step 5), checking the policies that may apply to the role of the job submitter (step 6), validating the user contextual information (step 7), and confirming the required access to satisfy database policy and constraints (step 8). Once the authorization decision has been made, the decision is executed in the enforcement steps (step 9). The first enforcement task is to grant the required access (step 10). A background process should then be started to check the expiry of the lease (step 11). There are a few utility processes (step 12) for managing the leases: lease renewal, execution of the job, permission cleanup and lease cleanup. The rest of this section describes the details of each of these three main functions of the DRLA protocol.



User_Authen(): verify user's account in the system
Lease_Athen(): verify the received lease querying the lease with job ID from the virtual space

Lease_cycle_check(): verify the lease period

Policy_check(): load and check the policy set for the corresponding role

Context_check(): load and check the contextual policy for the corresponding role

DB_check(): load and check the database policy and permission for the corresponding role

Grant(): grant and update the lease regarding the approved permission

Expiry_check(): a background process checks the expiry of the lease

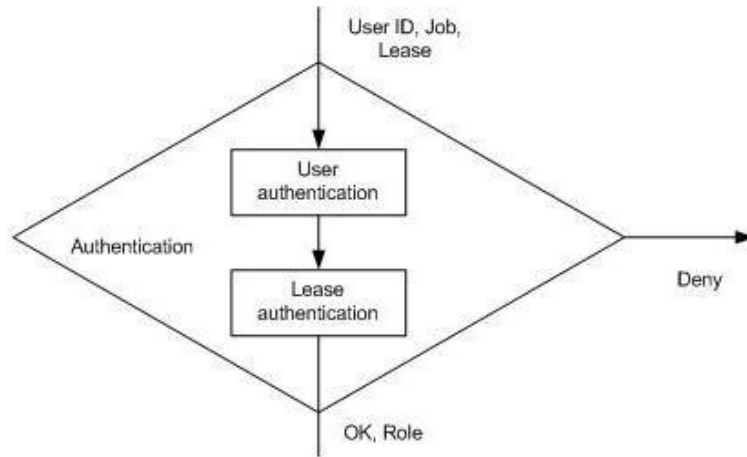
Lease_renew(): renew a lease

Execute_job(): run the job as specified in the job

Permission_cleanup(): a background process clean up expired permission

Lease_cleanup(): a background process clean up the permission information in a lease

Figure 3-9: DRLA grid service



1. Algorithm: User_Authen(User) // verify user's account in the system
2. If UserID exists and active in User table
3. then return okay
4. else return deny
5. end if

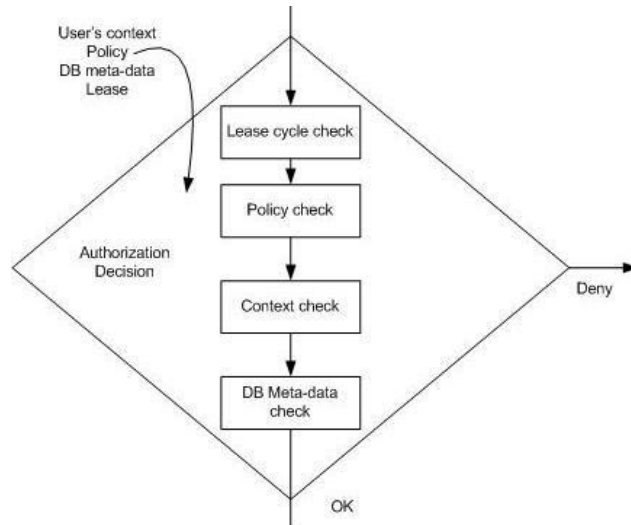
6. Algorithm: Lease_Athen(Job) // verify the received lease by querying the lease from the system
7. Query the Lease from the system
8. If the Lease exist in the system
9. then return okay
10. else return deny
11. end if

Figure 3-10: DRLA authentication and algorithms

The first step in a DRLA grid service is the authentication which confirms the user's registration record at the resource. The lease associated with a job will then be verified by querying the lease in the Grid. Each SN has recorded all leases that it issued and can be used to answer any lease verification request. This asserts that the received lease was issued by an SN, not by a hacker. In this thesis, we assume the SNs in a Grid are connected with a direct link and SNs are reliable and trustworthy, so the chance of any vulnerability between SNs is relatively low. Figure 3-10 depicts the authentication portion of the DRLA protocol. The authentication is divided in two main tasks: user authentication and lease authentication. User authentication (see algorithm, *User_Authen*, at line 1-5 in Figure 3-10) takes and verifies the user's account information at the

resource. Although, each user requires a valid login to access the Grid, it does not imply that the user has the right to use all resources in the Grid. A resource may only allow pre-registered grid users to access it. This process returns either approval or denial of the access. If user information is missed or any failure occurs at any point in the process, a denial decision takes precedence. Lease authentication (see algorithm, *Lease_Athen*, at line 6-11 in the Figure 3-10) verifies the received lease associated with a job by querying the lease in the Grid. All leases are generated by SNs in a Grid, so we can validate a lease by sending a validation request to other SNs. The query returns true to indicate that the provided lease can be found in the Grid, or false indicating the lease does not exist in the system. Based on the result, the process returns either an approval or denial as the authentication result. If a lease cannot be verified, no further action will be taken and the request will be denied.

Once the user and lease have been verified, the process moves to the next step: authorization. To authorize appropriate permissions for a user or role in a dynamic environment, we must consider the dynamic attributes of the users and the environment of the authorization process. The authorization decision is made based on two groups of dynamic attributes: users and resources. In such a dynamic setting, users' attributes may change frequently, such as location, time, and role. The policies for the resource could be changed from time to time due to changing business rules and requirements and may result in permission changes, so we must verify the permission at the database level.



1. Algorithm: Lease_cycle_check(Job) // verify the lease period
2. If lease period is still valid
3. then return okay
4. else return deny
5. end if

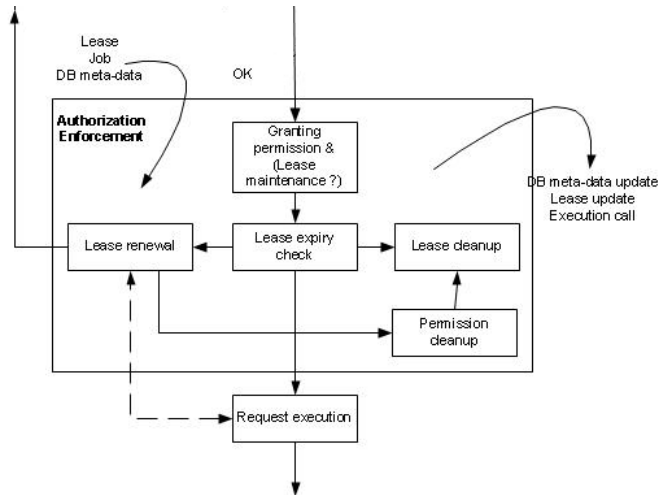
6. Algorithm: Policy_check(User) // load and check the policy set for the corresponding role
7. Load policies, PL, for the user
8. For each policy, P, in PL
9. if P is a policy set (with multiple sub-policy)
10. then if the user do not satisfy any sub-policy
11. then return deny
12. end if
13. else if the user do not satisfy the policy
14. then return deny
15. end if
16. end if
17. end for
18. return pass

19. Algorithm: Context_check(User) // load and check the contextual policy for the corresponding role
20. Load contextual policies, CPL, for the user
21. For each policy, P, in CPL
22. if the user do not satisfy the policy
23. then return deny
24. end if
25. end for
26. return pass

27. Algorithm: DB_check(User, Table) // load and check the db policy and permission for the role
28. Verify the database policy for the corresponding role and table
29. If the User in that Role is allowed to access the Table
30. then return okay
31. else return deny
32. end if

Figure 3-11: DRLA authorization and algorithms

Figure 3-11 shows the authorization processes in the DRLA protocol. One of the key goals of the DRLA authorization is to verify the dynamic factors in the system. Four main validations have been defined in this protocol: lease term, policy, contextual information, and database constraints. Any job submitted at the authorization stage has passed lease authentication. However, the lease term may affect the authorization decision. For example, a lease may expire very soon and be set to be a non-renewable lease, or the remaining time may be insufficient to perform the specified tasks, so there is no need to process the request further. The first step in the authorization is to check the lease terms (see the algorithm, *Lease_cycle_check*, at line 1-5 in the Figure 3-11), such as the renewability and the deadline of the job. As described earlier in this section, a policy could be changed to reflect the organization's security needs, so the second step is to validate the business policy defined in the system (see the algorithm, *Policy_check*, at line 6-18 in the Figure 3-11). In this thesis, the policy management (such as policy creation, conflict resolution, *etc.*) is not within the scope of the research so we assume that the policies are defined and managed by experts or a policy management system. Another important dynamic element is the user's contextual information. The contextual information includes the location, time, role, as well as other elements. Any change in the context may result in a different authorization decision (see the algorithm, *Context_check*, at line 19-26 in the Figure 3-11). The last dynamic component defined in the DRLA protocol is the database constraint. The changes on the database constraints may be caused by a change in business policies (see the algorithm, *DB_check*, at line 27-32 in the Figure 3-11). Any job must pass these validations or the authorization process denies the request.



1. Algorithm: Grant(User, Job, Table) // grant the approved permission to the role
2. If the required access to the Table is approved but not to the Role
3. then grant the access to the Role
4. end if
5. If the required access to the Table is already granted
6. then add the User and the Job into the job list at the resource
7. end if

8. Algorithm: Lease_renew(Job) // renew a lease
9. If Job is renewable
10. then if the associated lease is expiring in the predefined window
11. then if user pass the Context_check() and Policy_check()
12. then extend the lease period
13. else return deny
14. end if
15. else return no_renew_required_message
16. end if
17. else return not_renewable_message
18. end if

Figure 3-12: DRLA authentication enforcement and algorithms

Once the authorization has been approved, we enter the enforcement phase. Figure 3-12 illustrates the authorization enforcement processes and the main algorithms. Some domain specific tasks could be implemented in this phase. The requested permissions are granted and lease status is updated during the enforcement phase (see the algorithm, *Grant*, at line 1-5 in the Figure 3-12). Another main function in this phase is to renew a lease. If a user wants to maintain its role or privilege, it is required to renew the lease (see the algorithm, *Lease_renew*, at line 8-18 in the Figure 3-12). A background process is set

up to monitor (pull method) or listen (push method) any lease status change and triggers other supporting processes, such as lease renewal (including re-authorization), job execution and lease cleanup, once a status change has been detected.

III) Use Case

The scenario described in this section is used to illustrate the protocol. The access control protocol is implemented in a sales system of a store. In this system, there are only a few roles: New User (trainee), End User (sale associate), Manger, and Administrator. Each user is assigned at least one role. For a store, there are only three required types of required data: Sales transactions, Product, and Cost. For security reasons, not all users have access to all tables; some users may be allowed to access some data, but the privilege is not always granted, as they do not need access all the time. Head office specified a set of rules for granting the access privileges and storing data in the database. The following is a set of sample tables and records of the entities described above.

User: a list of users in the system

User ID	User Name
321456	Bob
159753	Tom
789321	Zoe

Context: a list of criteria and corresponding values that a policy may be used in the system

Context Name
Time
Subnet
Location

Role: a list of roles in the system

Role Name
Administrator
Manager
End User

User-Role: a list shows the assignments of users into roles

Role	User
New User	Anyone
End User	Zoe
Manger	Tom
Manger	Bob
Administrator	Bob

Table/Data: a list of tables/data available in the system

Table Name
Sales_Fact
Product_Dim
Cost_Fact

Approved Access: a list of approved access of each role

Table	Role	Status
Cost_Fact	End User	Approved
Product_Dim	End User	Granted
Product_Dim	New User	Approved
Sales_Fact	Manager	Granted

Policy: a list shows the access control policies of the roles based on the contexts

Policy ID	Context	Role	Min Value	Max Value	Depended Policy	Table Req.	Action
1	Time	End User	9	17			Allowed
2	Subnet	Manager	196.128.1.0	196.128.2.0	1		Allowed
3	Location	End User	PEI	PEI			Denied
4	Location	End User			1		Allowed
5	Location	New User	Web	Web	1	Product	Allowed

Now, we will illustrate the DRLA authorization process with a simple access request based on the sample records defined above. An end-user, Zoe, in a retail company from a store in Alberta (AB) has issued a request to retrieve the information of a product at 10am.

- Once the resource grid service receives Zoe's request, a lease is generated and tagged to the request. The grid service then forwards Zoe's account and lease information to DRLA grid service.
- The first phase of a DRLA grid service is authentication verification. The service authenticates the user's account and lease information. The first step is to verify that

Zoe is a valid user of the resource by checking its user database. Next, the grid service will verify the lease by sending a copy of the lease attached with the request to an SN. If the authentication function returns true, it indicates that the lease is valid and proceeds to the authorization phase. This verification is required for each request and each lease renewal request.

- In the authorization processes, it is required to check that the lease period specified in the lease is long enough to perform the request, or the lease must be renewable. Otherwise, the request should be rejected to ensure all requests could be completed within the leasing period. The next step is to enforce the policies. The service will verify user's contextual information and ensure it satisfies the policies. Once the verification is completed, the process can make a corresponding authorization decision. In this example, the following are Zoe's contextual information:

Contexts: Time = 10:00, Location = AB, Table = Product
Tables: Product
Decision: Approved

Since Zoe's contexts satisfy all policies, her request to retrieve product information will be granted. Therefore, the DRLA grid service has to perform a database policy check to confirm whether the authorized privilege has been granted, or not. After that, it brings the process to the next phase, authorization enforcement.

- The main purpose of the authorization enforcement phase is to enforce the approved access request. There are few major tasks in this phase: grant permission, check expiry, renew lease, execute job, cleanup permission and lease. Grant-permission is a function that records the decision and updates the lease with the corresponding privilege. Expiry-check constantly verifies the leases of the jobs that the resource has taken. If any job is required to renew its lease, this function will initiate a Lease-

renew function to renew the lease. During the lease renewal process, getting an updated user's context information becomes critical because the user's context could change often in a dynamic environment. If there is any expired lease, then Permission-cleanup and Lease-cleanup will be called to revoke the granted permission and remove the expired lease from the system. Execute-job function is used to execute the job specified in the request. The grid service of the resource will be notified about all actions in this phase, or status changes of the lease.

3.4. Summary

In this Chapter, we described the motivations and problems that DRLA addresses in Section 3.1. Then, we defined the Grid environment that will benefit from this research in Section 3.2 and illustrated the DRLA protocol in detail in Section 3.3. In summary, the proposed access control protocol, DRLA, provides a mechanism to grant the access required to dynamically access the required resources based on the user's context parameters. It satisfied the principle of least privilege by providing a more effective access control for a Grid. Most current so-called dynamic authorization models may do dynamic context verification and/or policy enforcement, and then granting pre-mapped or pre-assigned roles and permissions based on the result. The proposed protocol not only offers dynamic authorization but also context-awareness. The roles and permissions could be pre-granted, pre-approved, and/or context or policy-approved.

Chapter 4: Simulation with DRLA Protocol

This chapter describes the simulation architecture of an authorization system implemented with the DRLA protocol described in Chapter 3. The main purposes of this implementation are to demonstrate the operations of the DRLA protocol and to study its behaviors and performance in a simulated Grid environment. Another main goal is to illustrate and compare the new functionalities provided in the DRLA that are missed in the other models. This chapter is organized as follows: Section 4.1 reviews the simulation environment and configuration; Section 4.2 discusses the details of the simulation; and Section 4.3 summarizes the test cases defined for this experiment.

4.1. Simulation Environment

To illustrate the processes defined in the DRLA protocol in a simulated environment, a few assumptions and simplifications have been made to allow this implementation to focus on the main features of the protocol. First, all surrounding systems, protocols, and job management services connected to the simulated authorization system are assumed to be implemented and working properly. For example, these systems may include a job or resource scheduling system, job execution procedure, as well as networking protocols. Second, all information, including the locations and IP addresses, are programmed within the implementation so that extra procedures are not required to extract the information.

To demonstrate the feasibility and functionalities of the DRLA protocol, a set of simulations has been implemented with the GridSim [78] toolkit with Java 6. GridSim is a toolkit that allows us to model and simulate the entities (e.g. users, applications, resources, and resource brokers and schedulers) in either a parallel or distributed computing system, such as a Grid. This toolkit is commonly used to design and evaluate

protocols and algorithms in such systems. GridSim has been used to test many Grid algorithms and applications [78, 114, 115, 116, 117, 118, 119, 120] in a simulated Grid environment. All of these researches have proven that GridSim can provide a valid and effective simulated Grid environment for this research, and its API provides us flexibility to implement and demonstrate the new features of the DRLA protocol. The GridSim toolkit comes in three parts: SimJava, GridSim, and GridBroker. GridSim is built on SimJava which provides basic simulation features in a Java environment. All components in GridSim communicate with each other through message passing operations defined by SimJava. The most important component in the toolkit is GridSim, a Java API, which can be used to create a simulated Grid environment. The API contains classes which represent nodes, machines, I/O data package, as well as others. All of these classes can be extended to implement complex logics. This provides more flexibility for researchers to create simulated Grids to test algorithms. The hardware and software simulation environment used here are summarized as follows:

Hardware and software configuration:

Processor: AMD Phenom II X6, 2.6GHz

Memory: 8 GB

Operation System: MS Windows 7 64-bit

Programming Language: Java 6

Simulation Kit: GridSim Toolkit 5.2

4.2. Simulation Details

In our simulation, five main components have been implemented: *DRLA_GIS*, *DRLA_GridUser*, *DRLA_GridResource*, *DRLA_Service*, and *DRLA_Lease*. A key component in a GridSim environment is the Grid Information Service (GIS). A GIS is an entity that provides registration, indexing and discovery services for all other grid

entities. The grid resources announce their readiness to process requests by registering themselves with the GIS in the simulated environment. For this simulation, *DRLA_GIS* has been created and extended from the original GIS class in the GridSim API package to handle extra job management tasks that are needed. In this experiment, all the communication between entities and registration of the entities are managed by *DRLA_GIS*.

There are two main tasks that a grid user, a *DRLA_GridUser*, in this simulation must perform: maintaining its contextual information and initiating a request to a resource. To keep the job submission and testing method simple, each user will submit a job to all resources that are available in the simulated system. *DRLA_GridResource* defines the resource details (such as processing element, type and processing speed). *DRLA_GridResource* is also responsible for the main task in the DRLA protocol, which is to perform the dynamic role lease authorization.

Figure 4-1 shows the DRLA authorization process for a typical request from a user to a resource in the simulated environment. Before starting a communication between a user and a resource, both parties must register with the GIS so that the GIS can answer an enquiry about the availability of the resources in the system. Once a User receives the resource list, they can submit a job to the required Resource. Each request must be submitted with the submitter's (user) contextual information, such as the location or role, for the DRLA process. The Resource will create a lease (*DRLA_Lease*) for each received job before starting the DRLA process because the lease is an essential component in the authorization process. A Lease contains all leasing information (such as lease issued date, expiry, and renewability) required for the authorization process, and a lease may be

updated during the processing to record any status changes. Each submitted request must be authorized before the job is executed. Once the request is authorized, the Resource will execute the request and continue to monitor the user's contextual information as required in the DRLA protocol. This ensures that a user's dynamic change is not violating any Resource's policies; moreover, it guarantees any dynamic policy changes have been enforced on all new requests and jobs in progress. Any policy violation results in the job's abortion. The resource will notify the submitter regarding the request's completion and then the user can claim the result from the resource. The resource will also release the lease attached to the completed job.

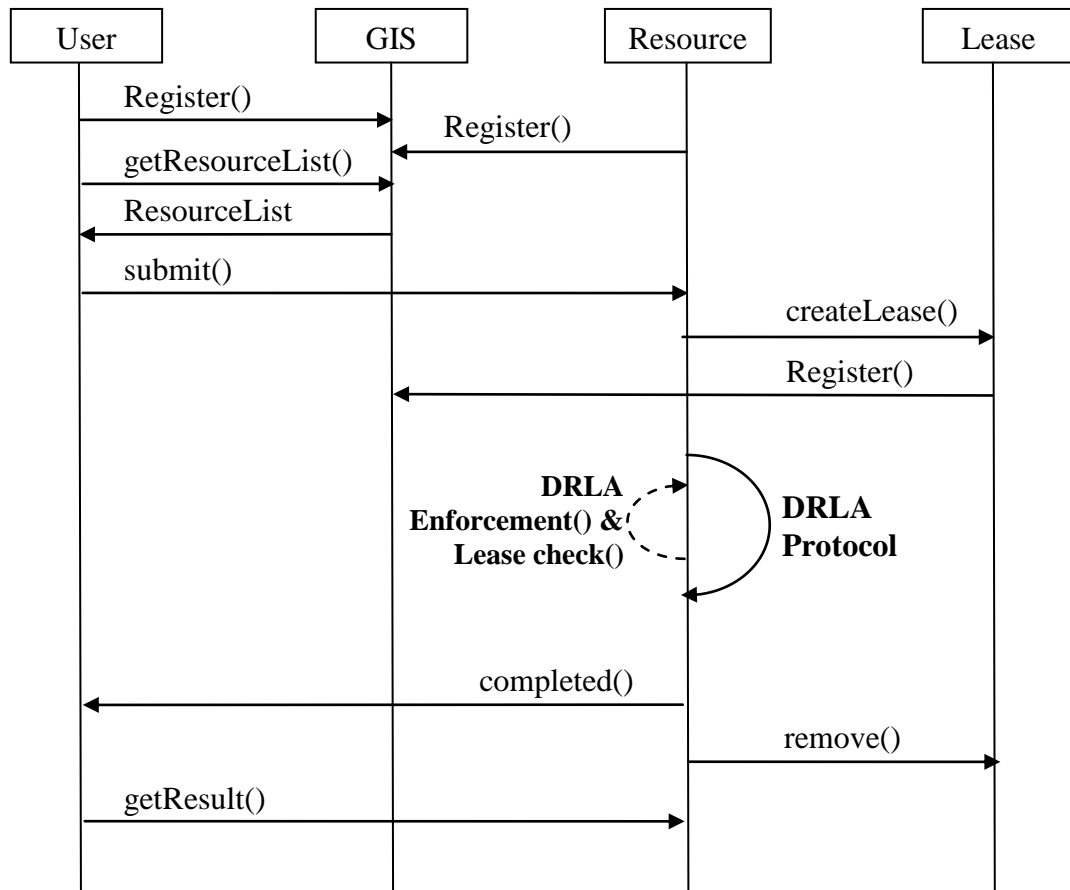
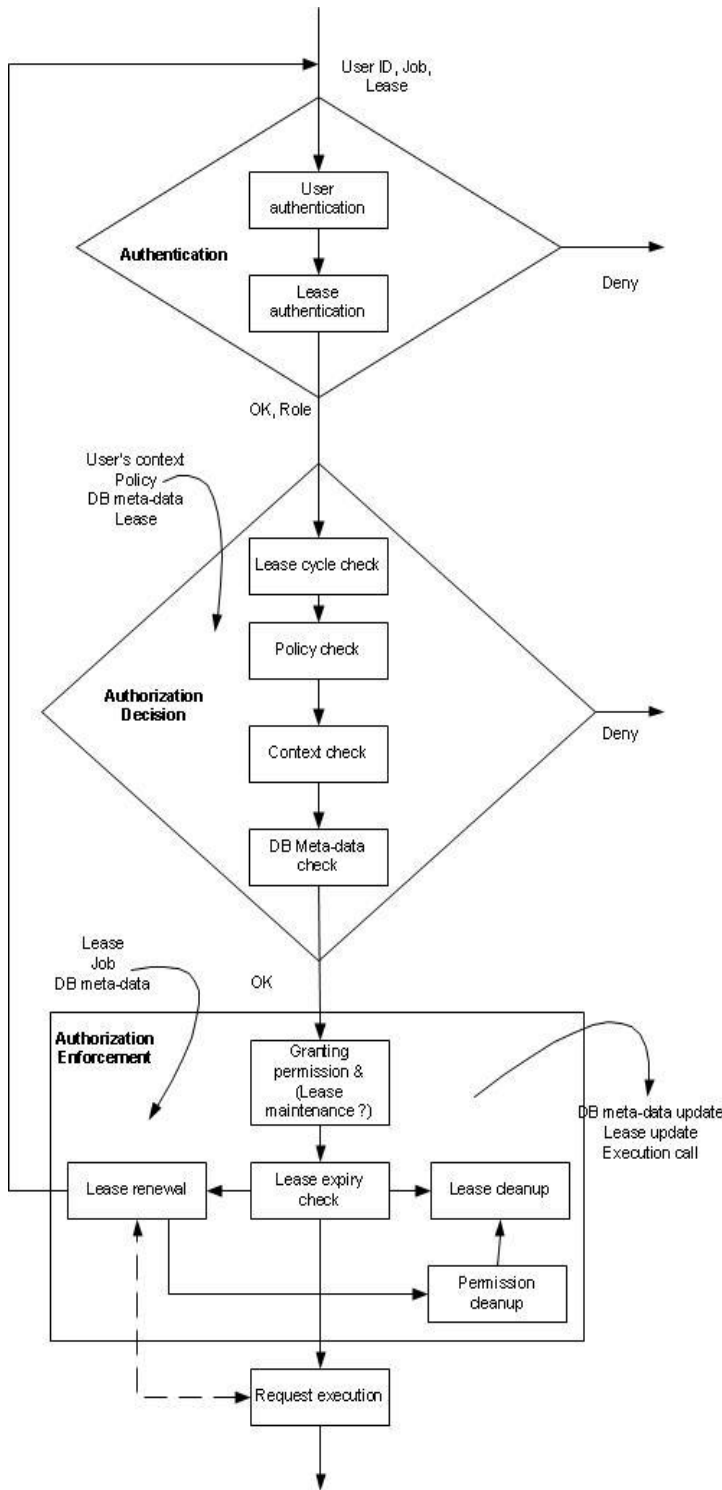


Figure 4-1: Sequence Diagram of a Request in the Simulated Environment



User_Authen(): verify user's account in the system

Lease_Athen(): verify the received lease querying the lease with job ID from the virtual space

Lease_cycle_check(): verify the lease period

Policy_check(): load and check the policy set for the corresponding role

Context_check(): load and check the contextual policy for the corresponding role

DB_check(): load and check the database policy and permission for the corresponding role

Grant(): grant and update the lease regarding the approved permission

Expiry_check(): a background process checks the expiry of the lease

Lease_renew(): renew a lease

Execute_job(): run the job as specified in the job

Permission_cleanup(): a background process clean up expired permission

Lease_cleanup(): a background process clean up the permission information in a lease

Figure 4-2: Flow Diagram for a DRLA Grid Service

The major DRLA functions (see Figure 4-2) have been discussed in Chapter 3, leaving this section to describe the main implementation details. To simplify the implementation, all of the lists for users, policies, and accesses are stored in flat files and loaded into the memory during the process.

In a Grid environment, there are many dynamic factors, so allowing resource providers to change their policies is critical. However, these policy changes may cause conflicts between different policies. The execution sequence of the policies may also be overlooked in some cases. There are many policy combinations which make policy management a very complicated research topic which is out of the scope of this thesis. Thus, we assume that a policy management expert will review and resolve any policy problems and conflicts that may occur. We have adopted a basic policy management technique, a “deny takes precedence” approach [79, 80] in our simulation. In this experiment, a User is required to satisfy all applicable policies; any violation of a policy results in a failure in the DRLA authorization process; hence, the request will be denied. This approach may not be the most efficient to resolve policy conflicts, but it is an effective and safe way to provide secured access control.

We have discussed the DRLA protocol and its algorithms for the major functions in Chapter 3, as well as the major implementation approach in this section. The next section will review the implementation details of the test cases for our experiments. These test cases cover the most representative situations and evaluate the performance of the DRLA implementation in different settings.

4.3. Test cases

In these experiments, four test cases have been designed and implemented to evaluate the correctness of the protocol with different network topologies and dynamic behaviors. The test cases match the scenarios discussed in Section 3.2 regarding dynamic factors and policy changes. In the rest of this section, U , r , and R are used to represent user, router and resource, respectively. In each case, we ran the simulation with a matrix of users (1, 10, 100, and 1000) and resources (2 and 10). To provide a controlled testing environment for each test, all user and policy lists, user and policy sequences, and the number of applications running on the testing computer are held constant.

The first test case is a fully connected error-free network topology (see Figure 4-3), which is a simplified version of the fully connected network described in Section 3.2. The main idea of a fully connected network is that each node in the network can communicate with any other node in a single-hop. Figure 4-3 shows that all users (U_n) are on one side of the network and connected to the same router, (r_1). In the same manner, all resources (R_n) are connected on the other side of the network and are connected to the router (r_2). There is only one communication path between the routers so each user can reach any resource in the same direct path.

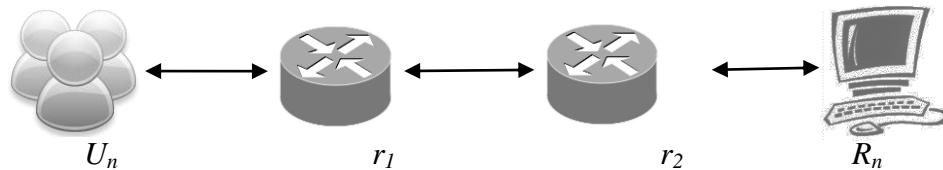


Figure 4-3: Fully connected network in the simulated environment

The second test case is a partially connected error-free network topology. The general idea of this scenario is that most communications in this kind of network are multi-hop, so the worst case scenario will be a request going through all nodes in the system to reach

the destination node. To create a stronger contrast for studying the difference between the fully and partially connected networks, we implement the worst case scenario in a simulated partially connected network. Figure 4-4 illustrates the network used to evaluate this scenario. In this example, ten resources are connected in sequence, and users are linked at one end of the network. The worst scenario for the users is to communicate with resource R_9 located at the other end of the network because there is only one communication path between them. In other words, any request from U_n to R_9 must pass through all nodes in the system.

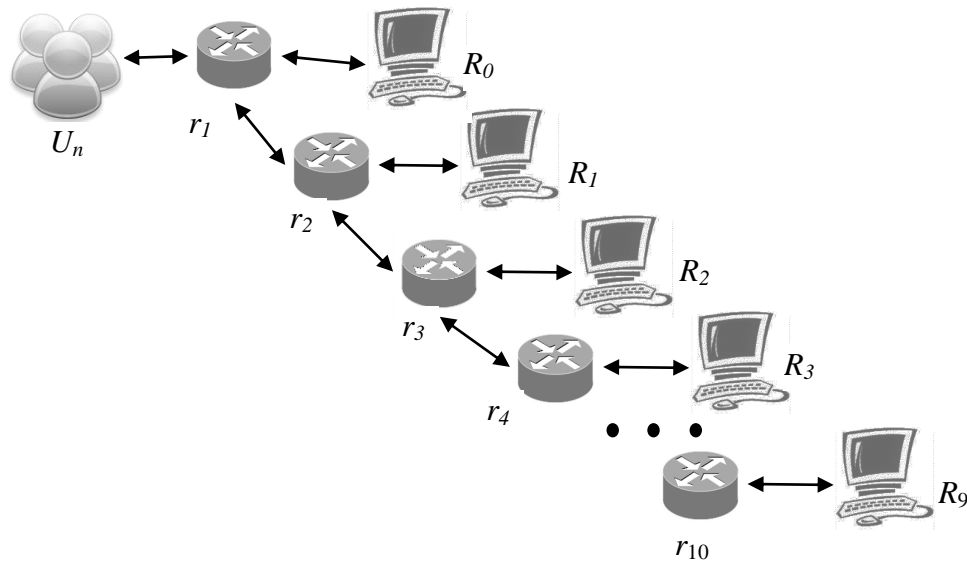


Figure 4-4: Partial connected network in the simulated environment

GridSim's Application Programming Interface (API) includes error generation for a simulated grid environment. In this implementation, we utilize the error generation of the API to facilitate our third test case for error handling and study the impact on the performance of the protocol. To demonstrate this error handling, we implement errors (or outage) on a resource. As specified in the DRLA protocol, all leased privileges or permissions will be revoked once the lease has expired. If an error occurs, the request

will be released back to the system when its lease expires, so the impact caused by the error is minimized and bounded within the lease period.

Dynamic change is one of the most important challenges that should be addressed in an authorization system. The last test case for the experiment studies the behavior of DRLA with dynamic user contextual information changes and the policy changes on the resource side. It also used to confirm that DRLA makes proper authorization results under those circumstances. User contextual changes are programmed in the experiment's control program and each change is embedded with a request. In the other words, the program is designed to send user's request multiple times, but each request is sent with different combinations of the contextual information to simulate the contextual changes. For example, each user sends two requests, one with role A from subnet B at time C and the other one with role A from subnet D at time C (another sample used in the simulation also shown in Figure 5-8). Another set of tests has been set up to demonstrate policy changes that may happen to a resource. The program is coded to execute the same request twice; each execution is processed with a different policy to simulate a policy change. For example, a resource provider may reduce the access hour. To simulate this change, we run the DRLA authorization process on a request with two policies, one with the old access hour and another one with the new hour (another sample used in the simulation also shown in Figure 5-12). In this way, we can test the impact of the different policies on the same request. The predefined request set will be executed in sequence to simulate the changes happening in a short period of time. This functional test has been performed in the two different network topologies as described in Figure 4-3 and Figure 4-4.

4.4. Summary

In this chapter, we reviewed the simulation environment and configuration in Section 4.1, and discussed the details of the simulation in Section 4.2. We also summarized the test cases defined for this implementation in Section 4.3. The main purposes of this implementation are to demonstrate the operations of the DRLA protocol and to study its behaviors and performance in a simulated Grid environment. Another main goal is to illustrate and compare the new functionalities provided in the DRLA that are missed in the other models.

Chapter 5: Simulation Results and Analysis

The testing environment and test cases have been described in Chapter 4. The testing results are summarized and analyzed in this chapter. As discussed earlier, there are four test cases, so the results are summarized in their corresponding areas: network topology differences, failure impacts, dynamic contextual changes and policy changes. Throughput (simulation time) is measured in each test (total time, total and average protocol execution time in the simulation). All results are presented in logarithmic to provide a more manageable graph size for the analysis.

Before getting into the detail of each case, a simulation output is used to illustrate the general steps of each main procedure of the proposed protocol. Figure 5-1 shows a sample simulation output from this implementation, with one user and two resources. Lines 20-27 show the authentication process, which includes user authentication and lease authentication steps. Lines 28-37 show the authorization process, which encapsulates lease-cycle-check, policy-check, contextual-check, and database-check procedures. Lines 38-48 and 69-79 show the enforcement process, including the tasks for monitoring lease changes, user contextual information changes, and job execution. Lines 92-95 show the completion of the simulation, and Lines 105-114 show the network topology that is created in the simulation.

```

-----Configuration: Version3 - JDK version 1.6.0_21 <Default> - <Default>-----
Exp. Main: Starting network example ...
1  Initialising...
2  Exp. Main: Initializing GridSim package at 20110515-18:48:21:760
3  Exp. Main: Starting to create one Grid resource with 3 Machines
4  Res_0: resource has been created with ID 5
5  Exp. Main: Created one Grid resource (name: Res_0 - id: 5)
6  Exp. Main: Res_0 is created
7  Exp. Main: Starting to create one Grid resource with 3 Machines
8  Res_1: resource has been created with ID 10
9  Exp. Main: Created one Grid resource (name: Res_1 - id: 10)
10 Exp. Main: Res_1 is created
11 User_1: user has been created with id = 15
12 Exp. Main: User_1 is created. Tom
13 Exp. Main: connect Thread[User_1,5,main] to the router, r1
14 Exp. Main: Simulation starts at 20110515-18:48:21:791
15 Starting GridSim version 5.0
16 Entities started.
17 User_1: Starting body() at 20110515-18:48:21:791
18 User_1: Received ResourceCharacteristics from Res_1(id = 10)
19 User_1: Sending JOB tag to Res_1 at 20110515-18:48:21:807
20 Res_1: received request from User_1 at 20110515-18:48:21:807
21 Res_1: DRLA process begin ...
22 Res_1: User_1 passed lease authentication
23 Res_1: User_1 starts authentication at 20110515-18:48:21:822
24 Res_1: User_1 is found in the control list.
25 Res_1: User_1 passed user authentication
26 Res_1: User_1 starts authorization at 20110515-18:48:21:822
27 Res_1: User_1 passed lease cycle check
28 Res_1: User_1 is found in the User-Role control list.
29 Res_1: User_1 starts approved_access at 20110515-18:48:21:822
30 Res_1: User_1 is found in the Approved_Access control list.
31 Res_1: User_1 passed policy check
32 Res_1: User_1 starts context_check at 20110515-18:48:21:822
33 Res_1: User_1 passed context check
34 Res_1: User_1 passed database check
35 Res_1: User_1's access request on Sales_Fact has been APPROVED !!!
36 Res_1: User_1 starts enforcement at time 20110515-18:48:21:822
37 Res_1: Starts DRLA_Enforcement for User_1 at time 20110515-18:48:21:822
38 Res_1: User_1's lease will be expire at: 20110515-18:48:26:807
39 Res_1: User_1's lease is renewable = true
40 Res_1: User_1's lease is still valid
41 Res_1: User_1's lease will be expire at: 20110515-18:48:26:807
42 Res_1: User_1's lease is renewable = true
43 Res_1: User_1's lease is still valid
44 Res_1: User_1's lease will be expire at: 20110515-18:48:26:807
45 Res_1: User_1's lease is renewable = true
46 Res_1: User_1's lease is still valid
47 User_1: Received ResourceCharacteristics from Res_0(id = 5)
48 User_1: Sending JOB tag to Res_0 at 20110515-18:48:21:822
49 User_1: Finished body() at time 20110515-18:48:21:822
50 User_1: Completed in 31 milliseconds
51 Res_0: received request from User_1 at 20110515-18:48:21:822
52 Res_0: DRLA process begin ...
53 Res_0: User_1 passed lease authentication
54 Res_0: User_1 starts authentication at 20110515-18:48:21:822
55 Res_0: User_1 is found in the control list.
56 Res_0: User_1 passed user authentication
57 Res_0: User_1 starts authorization at 20110515-18:48:21:822
58 Res_0: User_1 passed lease cycle check
59 Res_0: User_1 is found in the User-Role control list.
60 Res_0: User_1 starts approved_access at 20110515-18:48:21:822
61 Res_0: User_1 is found in the Approved_Access control list.
62 Res_0: User_1 passed policy check
63 Res_0: User_1 starts context_check at 20110515-18:48:21:822

```

Authentication

Authorization

Enforcement

```

64 Res_0: User_1 passed context check
65 Res_0: User_1 passed database check
66 Res_0: User_1's access request on Sales_Fact has been APPROVED !!!

67 Res_0: User_1 starts enforcement at time 20110515-18:48:21:822
68 Res_0: Starts DRLA_Enforcement for User_1 at time 20110515-18:48:21:822

69 Res_1: User_1's lease will be expire at: 20110515-18:48:26:822
70 Res_1: User_1's lease is renewable = true
71 Res_1: User_1's lease is still valid
72 Res_0: User_1's lease will be expire at: 20110515-18:48:26:822
73 Res_0: User_1's lease is renewable = true
74 Res_0: User_1's lease is still valid
75 Res_1: User_1's lease will be expire at: 20110515-18:48:26:822
76 Res_1: User_1's lease is renewable = true
77 Res_1: User_1's lease is still valid

... ..

78 Res_1: User_1's lease will be expire at: 20110515-18:48:26:822
79 Res_1: User_1's lease is renewable = true
80 Res_1: User_1's lease is still valid
81 Res_0: User_1's lease will be expire at: 20110515-18:48:26:822
82 Res_0: User_1's lease is renewable = true
83 Res_0: User_1's lease is still valid
84 Res_1: User_1's lease will be expire at: 20110515-18:48:26:822
85 Res_1: User_1's lease is renewable = true
86 Res_1: User_1's lease is still valid
87 Res_0: User_1's request has been completed at Sun May 15 18:48:23 MDT 2011
88 Res_0: Finished the request from User_1 at 20110515-18:48:23:866
89 Res_0: Completed User_1's request in 2044 milliseconds

90 Res_0: exit
91 Res_1: User_1's request has been completed at Sun May 15 18:48:23 MDT 2011
92 Res_1: Finished the request from User_1 at 20110515-18:48:23:866
93 Res_1: Completed User_1's request in 2059 milliseconds

94 Res_1: exit

95 User_1:%%% Exiting body()
96 DRLA_GIS: Notify all GridSim entities for shutting down.
97 Sim_system: No more future events
98 Gathering simulation data.
99 Simulation completed.
100 Exp. Main: Simulation ended at 20110515-18:48:23:991
101 Exp. Main: Simulation completed in 2200 milliseconds

102 --- Routing Table for router1 ---
103 router2          r1_r2_link
104 User_1          User_1_link
105 Res_1           router2
106 Res_0           router2
107 -----

108 --- Routing Table for router2 ---
109 Res_1           Res_1_link
110 Res_0           Res_0_link
111 router1        r1_r2_link
112 User_1         router1
113 -----

114 Exp. Main:
115 Finish DRLA experiment ...

116 Process completed.

```

Enforcement

Completion

Network connection

Figure 5-1: A Sample of the simulation Output

5.1. Analysis on Network Topology Difference

The first two test sets study the impact of network topologies on the proposed protocol. The network topologies that have been set up in this experiment are a fully-connected network and a partially connected network. As discussed in Section 4.3, a fully-connected network allows a user to reach any resource in single-hop. In contrast, a user and a resource may need to communicate in multi-hop fashion on a partially-connected network. The simulation results for the fully-connected network test case will be reviewed first, followed by the evaluation for the test case of the partially-connected network.

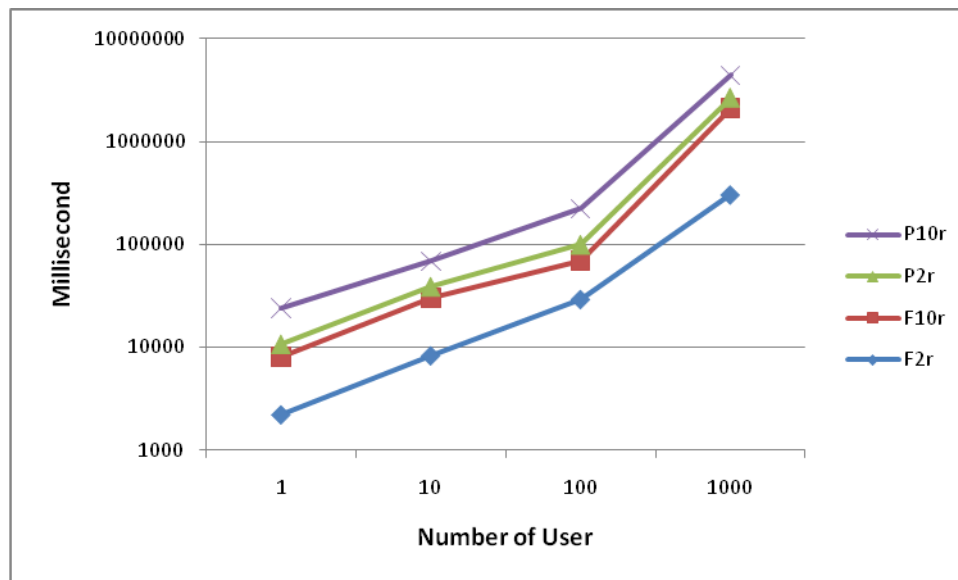


Figure 5-2: Total Simulation Time in an Error-free Environment

The total simulation time of the first test case is summarized in the Figure 5-2. The trends of each testing group (curve) are very similar. These results indicate that each group has a similar behavior regardless of the tested network topologies and the testing matrix. As expected, partially-connected networks (*P10r* and *P2r*) take longer than a fully-connected

network (*F10r* and *F2r*). In a partially-connected network, most communications are completed in multi-hop, so the communication costs increase. The partially-connected network in this experiment has been implemented as the worst case scenario (sequential network) so the result shows the lower bound of the performance. Another observation that is worthy to note, that the performance of *F10r* and *P2r* are very close. It indicates that a fully connected network may not always outperform than a partially connected network. The optimal performance depends on an optimal matrix of network, users and resources.

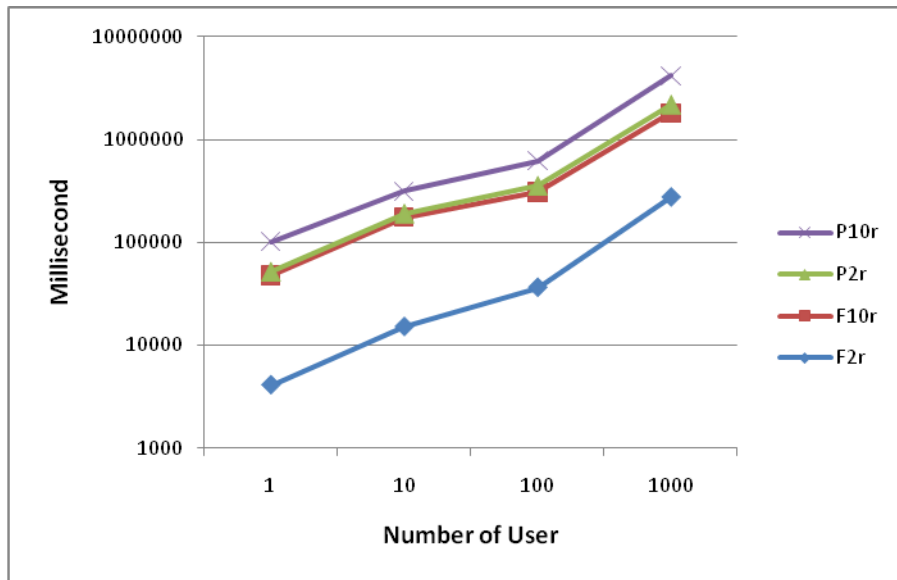


Figure 5-3: DRLA Protocol Total Execution Time in an Error-free Environment

Figure 5-2 shows the total simulation time required for each case, including end-to-end communication time required to initialize the simulation, register with the GIS, send messages between entities and the DRLA process, as well as the actual DRLA protocol execution. To have a better understanding regarding the behavior of the DRLA protocol, the total DRLA protocol execution time has also been recorded and reported in Figure 5-3 and the average execution time of each DRLA request is summarized in Figure 5-4.

The graphs show that the vulnerability of the tested networks is very high because the average time of the DRLA requests decrease as the number of requests increase. This implies that the number of requests that did not pass the DRLA authorization is much greater than the requests that satisfied the authorization criteria. Indeed, only 2.5% of the requests (225 out of 9000) have been defined with valid contextual information in this experiment. The graphs also show a persistent behavior across different testing sets between the two tested topologies. It confirms that there is no dynamic error occurred in the simulated environment and the time was taken that is purely for the essential operations of the process.

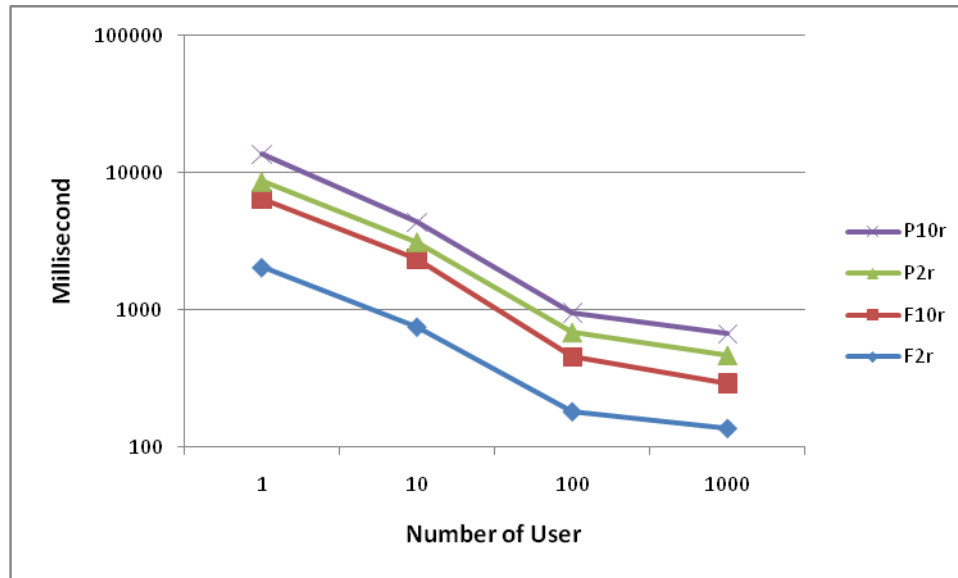


Figure 5-4: DRLA Protocol Average Execution Time in an Error-free Environment

5.2. Analysis on Non-error-free Environment

The previous section discussed DRLA behavior in error-free network topologies. The next case is extended from the previous ones but executed in a non-error-free environment, so this more closely resembles a real grid environment.

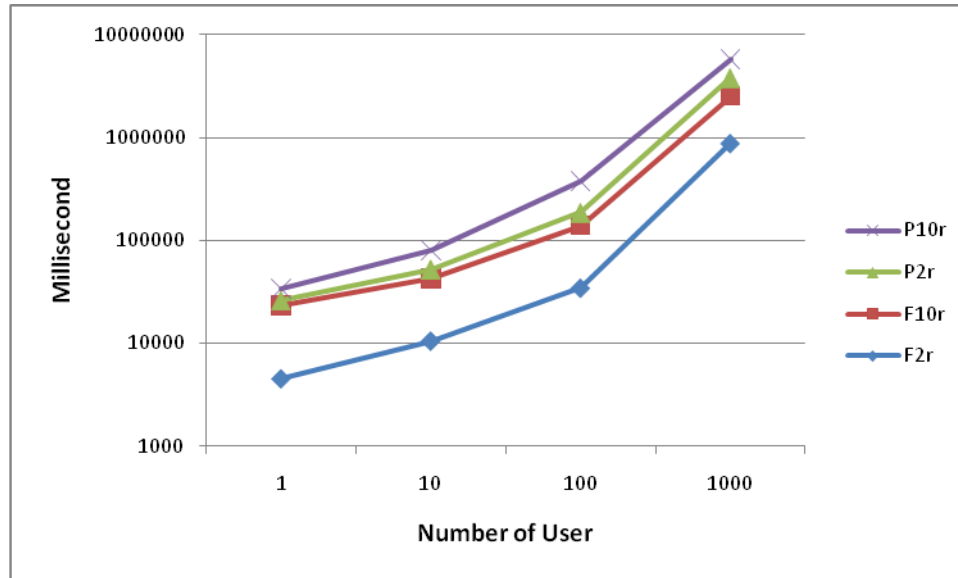


Figure 5-5: Total Simulation Time in a non-error-free Environment

Once again, there are three graphs to record the total simulation time (Figure 5-5), total DRLA execution time (Figure 5-6), and average DRLA execution time (Figure 5-7). By comparing the graphs, we realize that the simulation in the non-error-free environment not surprisingly takes more time than the error-free environment. This observation is expected because most errors in a network increase the number of re-sent messages; and the additional messages required for handling errors increase communication costs. However, the average and total DRLA execution time for a non-error-free environment is slightly lower than what is required for the error-free environment. The results show that the errors generated in the simulations caused at least one resource to become unavailable in the system. Therefore, the total and average DRLA execution time is slightly reduced, but this should not diminish the protocol performance because the error is generated in all test cases. The previous two observations suggest that the overhead for handling failures is the primary factor in the increased simulation time, and the performance of the DRLA protocol is very stable in the tested network topologies, even with different loads.

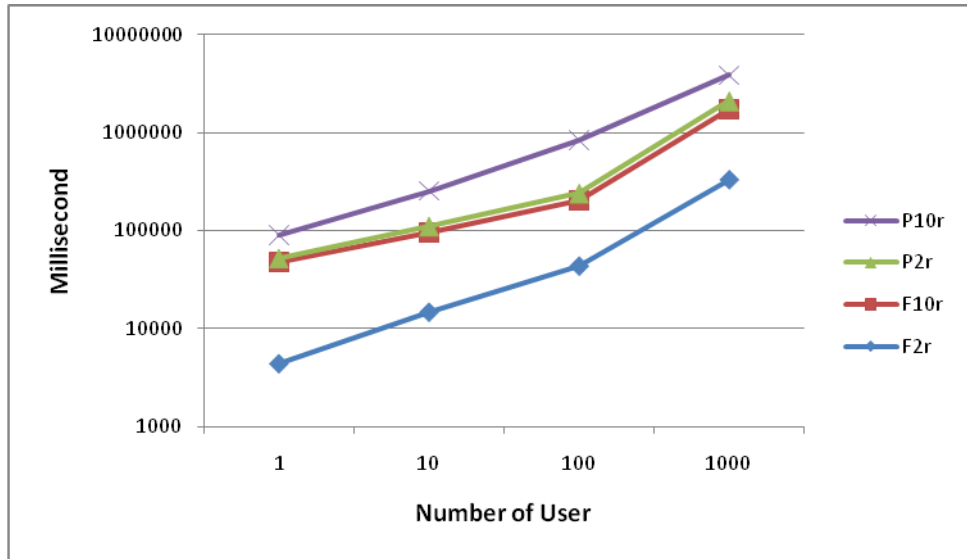


Figure 5-6: DRLA Protocol Execution Time in a Non-error-free Environment

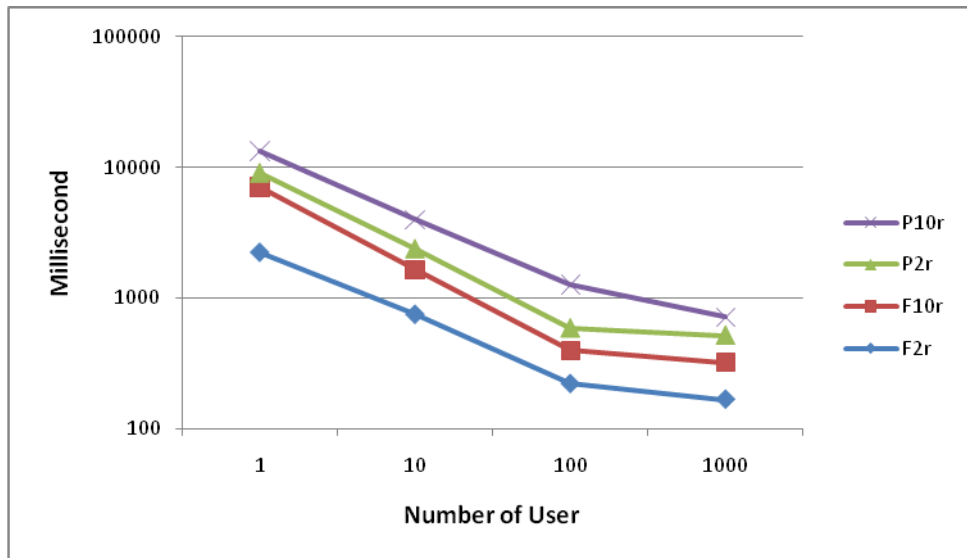


Figure 5-7: DRLA Protocol Ave. Execution Time in a Non-error-free Environment

Users' requests
 Role,User,Table,Time,Subnet,Loca
 Manager,Tom,Sales_Fact,6,1,0
 End User,Tom,Sales_Fact,6,1,0
 End User,Zoe,Employee,10,0,2
 End User,Zoe,Sales_Fact,10,0,2

Policies
 Policy ID,Context,Role,Min Value,Max Value,Depended Policy,Table Req.,Action
 1,Time,End User,9,17,, ,1
 2,Subnet,Manager,1,2,1, ,1
 3,Location,End User,1,1,, ,0
 4,Time,Manager,5,17,, ,1

Figure 5-8: Users' request and policy for contextual change testing

5.3. Analysis on Dynamic User Context & Policy Changes

To demonstrate the dynamic characteristics of the DRLA protocol, two sets of simulations have been setup in this experiment: (a) user contextual change and (b) policy change. The first set of tests shows how the DRLA protocol handles changes to a user's contextual information used in the authorization process. In the same manner, the second set of the tests is set up to demonstrate how policy changes are managed during the authorization process.

Figure 5-8 shows the predefined users' requests and policies, which are used to simulate user contextual change and resource policy change. There is only one set of policies defined in this test, so all requests will use the same parameters for their authorization. In the request set, the first two requests were defined for the same user but with different roles, and the next two requests are defined for another user, requesting different information. Figure 5-9, Figure 5-10 and Figure 5-11 show a sample of the simulation results. Figure 5-9 shows that all requests have been read into the simulation. In this experiment, each request is treated as an independent user without consideration for whether the requests are coming from the same user or not. As the example in Figure 5-10 and Figure 5-11 illustrate, the first two requests are generated by the same user but treated as two users in the simulation. Figure 5-10 and Figure 5-11 show the authorization results for the first two requests, which were generated by the same user. One of them had been denied during the authorization steps, and the other was approved due to the contextual difference in the roles used in the requests. This example shows the context changes have been captured and reflected in the authorization decision.

```
1 User_1: user has been created with id = 30
2 Exp. Main: User_1 is created. Tom
3 User_2: user has been created with id = 35
4 Exp. Main: User_2 is created. Tom
5 User_3: user has been created with id = 40
6 Exp. Main: User_3 is created. Zoe
7 User_4: user has been created with id = 45
8 Exp. Main: User_4 is created. Zoe
```

Figure 5-9: Sample output – request submission

```
9 Res_0: User_2 passed lease authentication
10 Res_0: User_2 starts authentication at 20110703-23:04:33:556
11 Res_0: User_2 is found in the control list.
12 Res_0: User_2 passed user authentication
13 Res_0: User_2 starts authorization at 20110703-23:04:33:556
14 Res_0: User_2 passed lease cycle check
15 Res_0: User_2 and the role are NOT found in the User-Role control list.
16 Res_0: User_2 starts approved_access at 20110703-23:04:33:556
17 Res_0: User_2 and the requested table, Sales_Fact, are NOT found in the Approved_Access control list.
18 Res_0: User_2 failed policy check
19 Res_0: User_2's access request on Sales_Fact has been denied !!!
20 Res_0: Finished the request from User_2 at 20110703-23:04:33:556
21 Res_0: Completed User_2's request in 0 milliseconds
```

Figure 5-10: Sample output – authorization result for the first request

```
22 Res_0: received request from User_1 at 20110703-23:04:33:602
23 Res_0: DRLA process begin ... 1000
24 Res_0: User_1 passed lease authentication
25 Res_0: User_1 starts authentication at 20110703-23:04:33:602
26 Res_0: User_1 is found in the control list.
27 Res_0: User_1 passed user authentication
28 Res_0: User_1 starts authorization at 20110703-23:04:33:602
29 Res_0: User_1 passed lease cycle check
30 Res_0: User_1 is found in the User-Role control list.
31 Res_0: User_1 starts approved_access at 20110703-23:04:33:602
32 Res_0: User_1 is found in the Approved_Access control list.
33 Res_0: User_1 passed policy check
34 Res_0: User_1 starts context_check at 20110703-23:04:33:665
35 Res_0: User_1 passed context check
36 Res_0: User_1 passed database check
37 Res_0: User_1's access request on Sales_Fact has been APPROVED !!!
38 Res_0: User_1 starts enforcement at time 20110703-23:04:33:665
39 Res_0: Starts DRLA_Enforcement for User_1 at time 20110703-23:04:33:665
```

Figure 5-11: Sample output – authorization result for the second request

Another set of tests (see Figure 5-12, Figure 5-13 and Figure 5-14) were used to demonstrate how the DRLA protocol handles dynamic policy changes. Figure 5-12 shows the user requests and policies that have been defined for this experiment. To simulate dynamic policy changes, two sets of policies have been set up, as shown in Figure 5-12. The policies were chosen in the main control program using the round-robin technique. A user sent in multiple requests to the same resource; each request had been read into the simulation and recoded as an independent request, however, they were handled with different policies. Figure 5-13 shows the request was processed with the

policy set #1 and the request was authorized. However, the same request was denied with the policy set #2, as shown in Figure 5-14. This example illustrates the capability of DRLA to handle the policy changes.

Users' requests

Role,User,Table,Time,Subnet,Loca

Manager,Tom,Sales_Fact,6,1,0
 Manager,Tom,Sales_Fact,6,1,0

Policy set #1

Policy ID,Context,Role,Min Value,Max Value,Depended Policy,Table Req.,Action
 1,Time,End User,9,17, , ,1
 2,Subnet,Manager,1,2,1, ,1
 3,Location,End User,1,1, , ,0
 4,Time,Manager,5,17, , ,1

Policy set #2

Policy ID,Context,Role,Min Value,Max Value,Depended Policy,Table Req.,Action
 1,Time,End User,12,17, , ,1
 2,Subnet,Manager,1,2,1, ,1
 3,Location,End User,1,1, , ,0
 4,Time,Manager,9,17, , ,1

Figure 5-12: Sample Users' request and policy for policy change testing

```

1 Res_0: User_2 passed lease authentication
2 Res_0: User_2 starts authentication at 20110705-04:50:55:519
3 Res_0: User_2 is found in the control list.
4 Res_0: User_2 passed user authentication
5 Res_0: User_2 starts authorization at 20110705-04:50:55:519
6 Res_0: User_2 passed lease cycle check
7 Res_0: User_2 is found in the User-Role control list.
8 Res_0: User_2 starts approved_access at 20110705-04:50:55:519
9 Res_0: User_2 is found in the Approved_Access control list.
10 Res_0: User_2 passed policy check
11 Res_0: User_2 starts context_check at 20110705-04:50:55:519
12 Res_0: User_2 passed context check
13 Res_0: User_2 passed database check
14 Res_0: User_2's access request on Sales_Fact has been APPROVED !!!
15 Res_0: User_2 starts enforcement at time 20110705-04:50:55:519
16 Res_0: Starts DRLA_Enforcement for User_2 at time 20110705-04:50:55:519
  
```

Figure 5-13: Sample output – Policy set #1

```

17 Res_0: User_1 starts authentication at 20110705-04:51:00:901
18 Res_0: User_1 is found in the control list.
19 Res_0: User_1 passed user authentication
20 Res_0: User_1 starts authorization at 20110705-04:51:00:963
21 Res_0: User_1 passed lease cycle check
22 Res_0: User_1 is found in the User-Role control list.
23 Res_0: User_1 starts approved_access at 20110705-04:51:00:963
24 Res_0: User_1 is found in the Approved_Access control list.
25 Res_0: User_1 passed policy check
26 Res_0: User_1 starts context_check at 20110705-04:51:00:963
27 Res_0: User_1 failed context check
28 Res_0: User_1's access request on Sales_Fact has been denied !!!
29 Res_0: Finished the request from User_1 at 20110705-04:51:01:025
30 Res_0: Completed User_1's request in 124 milliseconds
  
```

Figure 5-14: Sample output – Policy set #2

5.4. Analysis with other Authorization models

The most significant features provided by the DRLA protocol are the dynamic user's contextual information verification along with the lease structure to address the "long holding permission" problem. The detail of the protocol has been discussed in Chapter 3. The performance of the protocol in an error-free and a non error-free environment have been discussed in Sections 5.1 and 5.2. The research results also illustrate how the protocol handles the user's contextual changes and resource policy changes in Section 5.3. The simulation results from sections 5.1 to 5.3 show that there is no significant delay caused by the authorization processes and correct authorization decisions had been produced. In this section, we will use the simulation results to illustrate the differences between DRLA and other authorization models. As described in Chapter 2, in this study the reviewed authorization models have been categorized into two groups, the *dynamic context aware authorization models* and the *active/dynamic authorization models*. The models in each group share similar characteristics and functions. The *dynamic context aware authorization models* group focuses on user context verification and the *active/dynamic authorization models* group addresses real-time authorization. To manifest the benefits of the DRLA's new functions, we have implemented CAAC [6, 9] and DRBAC [7, 8] models (from those two groups, respectively) in a simulated environment to represent the behaviors for the models in these two groups.

Figure 5-15 shows a basic case in which a user sends in a request (lines 3-7) that passes through the DRLA process (lines 8-26). During the enforcement phase, the user's contextual information and lease status have been monitored until the request was completed. Figure 5-16 shows how DRLA handles a role change in the simulation. Lines 1-40 show the request submission and the DRLA authorization. During the enforcement

phase, once a change on a user's role is detected (line 42), the new role is verified against the policies and a new authorization decision is made. In this example, the new role is denied and the job is aborted (see lines 43-54). Similarly, any change in a defined context is verified when it occurs. Figure 5-17 shows that the user moved from a trusted subnet to a non-trusted subnet (line 20) which is reflected in the authorization decision (line 23-37).

```

1  Starting GridSim version 5.0
2  Entities started.
3  User_1: Starting body() at 20120315-22:48:58:113
4  User_1: Received ResourceCharacteristics from Res_1(id = 10)
5  User_1: Sending JOB tag to Res_1 at 20120315-22:48:58:124
6
7  Res_1: received request from User_1 at 20120315-22:48:58:133
8  Res_1: DRLA process begin ...
9  Res_1: User_1 passed lease authentication
10
11 Res_1: User_1 starts authentication at 20120315-22:48:58:133
12 Res_1: User_1 is found in the control list.
13 Res_1: User_1 passed user authentication
14
15 Res_1: User_1 starts authorization at 20120315-22:48:58:133
16 Res_1: User_1 passed lease cycle check
17 Res_1: User_1 is found in the User-Role control list.
18
19 Res_1: User_1 starts approved_access at 20120315-22:48:58:133
20 Res_1: User_1 is found in the Approved_Access control list.
21 Res_1: User_1 passed policy check
22
23 Res_1: User_1 starts content_check at 20120315-22:48:58:134
24 Res_1: User_1 passed content check
25 Res_1: User_1 passed database check
26 Res_1: User_1's access request on Sales_Fact has been APPROVED !!!
27
28 Res_1: User_1 starts enforcement at time 20120315-22:48:58:134
29 Res_1: Starts DRLA_Enforcement for User_1 at time 20120315-22:48:58:134
30
31 Res_1: User_1 is found in the User-Role control list.
32
33 *****
34 Res_1 20120315-22:48:58:134 : User_1's policy check = true
35 *****
36
37 Res_1: User_1's lease will be expire at: 20120315-22:48:59:333
38 Res_1: User_1's lease is renewable = true
39 Res_1: User_1's lease is still valid
40
... ..

40 *****
41 Res_1 20120315-22:48:58:505 : User_1's policy check = true
42 *****
43
44 Res_1: User_1's lease will be expire at: 20120315-22:48:59:555
45 Res_1: User_1's lease is renewable = true
46 Res_0: Finished the request from User_1 at 20120315-22:48:58:555
47 Res_0: Completed User_1's request in 409 milliseconds
48
49 Res_0: exit
50 Res_1: Finished the request from User_1 at 20120315-22:48:58:555
51 Res_1: Completed User_1's request in 422 milliseconds
52
53 Res_1: exit
54 User_1: GridSim time 100012.15462857143

```

Figure 5-15: DRLA Experiment without Role Change

```

1 Starting GridSim version 5.0
2 Entities started.
3 User_1: Starting body() at 20120315-22:30:20:416
4 User_1: Received ResourceCharacteristics from Res_0(id = 5)
5 User_1: Sending JOB tag to Res_0 at 20120315-22:30:20:428
6
7 Res_0: received request from User_1 at 20120315-22:30:20:478
8 Res_0: DRLA process begin ...
9 Res_0: User_1 passed lease authentication
10
11 Res_0: User_1 starts authentication at 20120315-22:30:20:478
12 Res_0: User_1 is found in the control list.
13 Res_0: User_1 passed user authentication
14
15 Res_0: User_1 starts authorization at 20120315-22:30:20:478
16 Res_0: User_1 passed lease cycle check
17 Res_0: User_1 is found in the User-Role control list.
18
19 Res_0: User_1 starts approved_access at 20120315-22:30:20:479
20 Res_0: User_1 is found in the Approved_Access control list.
21 Res_0: User_1 passed policy check
22
23 Res_0: User_1 starts content_check at 20120315-22:30:20:479
24 Res_0: User_1 passed content check
25 Res_0: User_1 passed database check
26 Res_0: User_1's access request on Sales_Fact has been APPROVED !!!
27
28 Res_0: User_1 starts enforcement at time 20120315-22:30:20:479
29 Res_0: Starts DRLA_Enforcement for User_1 at time 20120315-22:30:20:479
30
31 Res_0: User_1 is found in the User-Role control list.
32
33 *****
34 Res_0 20120315-22:30:20:480 : User_1's policy check = true
35 *****
36
37 Res_0: User_1's lease will be expire at: 20120315-22:30:21:678
38 Res_0: User_1's lease is renewable = true
39 Res_0: User_1's lease is still valid
40 Res_0: User_1 is found in the User-Role control list.
41
42 ... ..
43 *****
44 User_1 changed role at 20120315-22:30:20:787
45 *****
46 Res_0: User_1 and the role are NOT found in the User-Role control list.
47 *****
48 Res_0 20120315-22:30:20:787 : User_1's policy check = false
49 *****
50
51 Res_0: Finished the request from User_1 at 20120315-22:30:20:787
52 Res_0: Completed User_1's request in 309 milliseconds
53
54 Res_0: exit
55 User_1: Gridsim time 100012.15462857143

```

Figure 5-16: DRLA Experiment with Role Change

```

1 *****
2 Res_1 20120315-23:33:30:997 : User_1's policy check = true
3 *****
4
5 Res_1: User_1's lease will be expire at: 20120315-23:33:35:737
6 Res_1: User_1's lease is renewable = true
7 Res_1: User_1's lease is still valid
8
9 Res_0: User_1 starts content_check at 20120315-23:33:31:046
10
11 *****
12 Res_0 20120315-23:33:31:046 : User_1's policy check = true
13 *****
14
15 Res_0: User_1's lease will be expire at: 20120315-23:33:35:737
16 Res_0: User_1's lease is renewable = true
17 Res_0: User_1's lease is still valid
18
19 *****
20 User_1 changed Subnet at 20120315-23:33:31:047
21 *****
22
23 Res_1: User_1 starts content_check at 20120315-23:33:31:047
24
25 *****
26 Res_1 20120315-23:33:31:047 : User_1's policy check = false
27 *****
28
29 Res_1: Finished the request from User_1 at 20120315-23:33:31:096
30 Res_1: Completed User_1's request in 375 milliseconds
31
32 Res_1: exit
33
34 Res_0: User_1 starts content_check at 20120315-23:33:31:096
35
36 *****
37 Res_0 20120315-23:33:31:097 : User_1's policy check = false
38 *****
39
40 Res_0: Finished the request from User_1 at 20120315-23:33:31:097
41 Res_0: Completed User_1's request in 360 milliseconds
42
43 Res_0: exit
44 User_1: GridSim time 100012.15462857143

```

Figure 5-17: DRLA Experiment with Context Change

The Context-Aware Access Control (CAAC) model [6] extended the traditional RBAC model and included context verification as an extra authorization criterion. This approach makes its authorization decisions based on the contexts at runtime, rather than simply the role of the user. However, the user's contextual information could be changing frequently in a dynamic environment. This implies that the contextual changes are not reflected in the authorization decision. Figure 5-18 shows a simulation that implemented the CAAC access control logic. Lines 1-9 indicate the user passed the authentication and role-based authorization criteria. Lines 12-24 show that the user passed the contextual information verification and gained access to the required data. Subsequently, the user's role was

changed (line 26); however, the change was not captured and reflected on the authorization decision. In other words, the business logic and rules are only enforced once at the initial authorization. The granted access remains active (turned on) until the end of the user's session; regardless the contextual changes. The dynamic factors in the environment could be changed at any time, so continuous monitoring should be considered in such a dynamic environment. The DRLA protocol proposed in this thesis uses leasing structure to handle the concerns mentioned above. Every granted access is attached to a lease, which is only valid within the lease period. The lease can be renewed before its expiry, but the contextual information is verified again during the renewal process. Any permission associated with an expired lease will be revoked. This approach provides continuous monitoring on the user's contextual information and addresses the long holding permission problem discussed in earlier chapters.

```

1  Res_1: received request from User_1 at 20120315-22:55:40:335
2  Res_1: CAAC process begin ...
3
4  Res_1: User_1 starts authentication at 20120315-22:55:40:335
5  Res_1: User_1 is found in the control list.
6  Res_1: User_1 passed user authentication
7
8  Res_1: User_1 starts authorization at 20120315-22:55:40:335
9  Res_1: User_1 is found in the User-Role control list.
10 Res_1: User_1 passed policy check
11
12 Res_1: User_1 starts content_check at 20120315-22:55:40:335
13 Res_1: User_1 passed content check
14 Res_0 20120315-22:55:40:338 : User_1 with Manager role, access to Sales_Fact still valid
15 Res_1 20120315-22:55:40:338 : User_1 with Manager role, access to Sales_Fact still valid
16 Res_0 20120315-22:55:40:338 : User_1 with Manager role, access to Sales_Fact still valid
17 Res_1 20120315-22:55:40:338 : User_1 with Manager role, access to Sales_Fact still valid
18 Res_0 20120315-22:55:40:338 : User_1 with Manager role, access to Sales_Fact still valid
19 Res_1 20120315-22:55:40:339 : User_1 with Manager role, access to Sales_Fact still valid
20 Res_0 20120315-22:55:40:339 : User_1 with Manager role, access to Sales_Fact still valid
21 Res_1 20120315-22:55:40:339 : User_1 with Manager role, access to Sales_Fact still valid
22 Res_0 20120315-22:55:40:339 : User_1 with Manager role, access to Sales_Fact still valid
23 Res_1 20120315-22:55:40:339 : User_1 with Manager role, access to Sales_Fact still valid
24 Res_0 20120315-22:55:40:340 : User_1 with Manager role, access to Sales_Fact still valid
25
26 Res_0 20120315-22:55:40:340 : User_1's role changed
27
28 Res_1 20120315-22:55:40:340 : User_1 with demoted role, access to Sales_Fact still valid
29 Res_0 20120315-22:55:40:340 : User_1 with demoted role, access to Sales_Fact still valid
30 Res_1 20120315-22:55:40:340 : User_1 with demoted role, access to Sales_Fact still valid
31 Res_0 20120315-22:55:40:340 : User_1 with demoted role, access to Sales_Fact still valid
32 Res_1 20120315-22:55:40:341 : User_1 with demoted role, access to Sales_Fact still valid
33 Res_0 20120315-22:55:40:341 : User_1 with demoted role, access to Sales_Fact still valid
34 Res_1 20120315-22:55:40:341 : User_1 with demoted role, access to Sales_Fact still valid

```

Figure 5-18: CAAC Implementation

```

1 Res_0: received request from User_1 at 20120317-10:54:24:141
2 Res_0: DRBAC process begin ...
3
4 Res_0: User_1 starts authentication at 20120317-10:54:24:141
5 Res_0: User_1 is found in the control list.
6 Res_0: User_1 passed user authentication
7
8 Res_0: User_1 starts authorization at 20120317-10:54:24:141
9 Res_0: User_1 passed lease cycle check
10 Res_0: User_1 is found in the User-Role control list.
11 Res_0: User_1 passed policy check
12
13 Res_0: User_1 starts content_check at 20120317-10:54:24:141
14 Res_0: User_1 passed content check
15 Res_1 20120317-10:54:24:142 : User_1 with Manager role, access to Sales_Fact still valid
16 Res_0 20120317-10:54:24:142 : User_1 with Manager role, access to Sales_Fact still valid
17 Res_1 20120317-10:54:24:142 : User_1 with Manager role, access to Sales_Fact still valid
18 Res_0 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
19 Res_1 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
20 Res_0 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
21 Res_1 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
22 Res_0 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
23 Res_1 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
24 Res_0 20120317-10:54:24:143 : User_1 with Manager role, access to Sales_Fact still valid
25 Res_1 20120317-10:54:24:144 : User_1 with Manager role, access to Sales_Fact still valid
26
27 Res_1 20120317-10:54:24:144 : User_1 changed Subnet
28
29 Res_0 20120317-10:54:24:144 : User_1 with Manager role, access to Sales_Fact still valid
30 Res_1 20120317-10:54:24:144 : User_1 with Manager role, access to Sales_Fact still valid
31 Res_0 20120317-10:54:24:144 : User_1 with Manager role, access to Sales_Fact still valid

```

Figure 5-19: DRBAC Implementation

The Dynamic Role Based Access Control (DRBAC) model [8, 7] handles the dynamic changes by updating the Subject-Roles relationship (Role Assignment) and the Roles-Permission relationship (Permission Assignment); hence, DRBAC reflects the role hierarchy for the users. However, this approach may not be appropriate all the time. In some cases, the parameters for the user's contextual changes are not necessarily causing a change in the relationship of the Roles-Subjects or Roles-Permissions. For example, a policy may specify that a user may only be allowed to access the system from subnet A. If a user sends in a request from subnet B, the user still satisfies the role and permission assignments, and gains access under the DRBAC scheme. An example of this simulation is shown in Figure 5-19. Lines 1-25 illustrate that the user passed both the authorization and the requirements of role assignment and permission assignment. The user moved from a trusted subnet to a non-trusted subnet (Line 27). This change did not affect the assignments defined under the DRBAC scheme, and in this simulation the user continues

to be able to access the information. However, the contextual information should always be reflected in the authorization decision. Under DRLA protocol, a role and associated permission are only leased to a user within a leasing period, and the user must maintain the contextual status. All permissions will be revoked once the lease has expired. A lease could be renewed, but all contextual information has to be verified again during the renewal process (as demonstrated in the Figure 5-16 and Figure 5-17). The DRLA utilizes the leasing structure to automatically un-bound any invalid user, role, or permission assignment.

The simulation is used to demonstrate the new functionalities provided by DRLA and compare to other models. Since the proposed functions are new, the performance comparison between DRLA and other models is irrelevant for this purpose, and therefore performance comparison is excluded from this study. The remainder of this chapter summarizes the functional differences between DRLA and a few other models reviewed in this thesis.

The Dynamic Authorization Management model [43] is a task-centric model that decomposes a task into smaller sub-tasks, assigning them to the roles in the system. The dynamic characteristics of this model allow it to efficiently decompose a task and dynamically build the Role-Task relationship. The dynamic contextual information of a user has not been considered in this authorization model, as it is designed for project management within an organization. In contrast, DRLA takes account of the change in users' contextual parameters. In a Grid, handling the dynamic user contextual

information changes becomes critical in the authorization process. DRLA uses the leasing approach to handle the dynamic changes, while minimizing the impact.

The active authorization management model [32] has considered the relationship between Roles and Users, and Roles and Permissions. Therefore, it introduced business rules (policies) and contextual parameters to represent the relationships, and these are the keys to active authorization. However, there is a “multi-domain environment that have not been discussed or addressed in this active authorization model. The dynamic nature of the contextual information is an important factor for making authorization decisions in such an environment. The authorization model has taken into account the contextual information to make the authorization decision. However, only considering the context during the authorization process is not enough to satisfy the security requirements for a highly heterogeneous and volatile environment.

Dynamically Administrated Role-Based Access Control (DARBAC) [34, 35] defined the entities (User, Role, Permission, Mission and Objective) and the static relationships or assignments between the entities at the build-time. The dynamic characteristics in the model are the user-to-role activation, user-to-missions participation and objectives-to-missions binding for a workflow management system. The activation, participation, and binding are directly or indirectly defined through the static entities because most of these relationships are defined in built time. Consequently, this model does not address the dynamic user context changes; contain a mechanism to change the assignments dynamically; or handle the dynamic contextual information or factors. Whereas DRLA is

designed to precisely satisfy these requirements, from the perspective of the dynamic changes.

Dynamic Authorization Framework for Multiple Authorization Types (DAFMAT) [16] focuses on the context of an application system rather than the operation system. In addition, it defines three authorization types (emergency, context and non-context) and their corresponding processes. However, defining some well-represented authorization types and processes that are suitable for most requests could be very difficult. Furthermore, covering all authorization scenarios efficiently with only three general authorization types is extremely complicated. The DRLA protocol is designed to cope with both application and operating systems; all the business rules could be mapped in a policy. Unlike the DAFMAT, DRLA not only verifies the user's contextual information during the authorization process, it also monitors the changes on that information and reflects the changes in the next lease renewal process.

Task-Based Access Control (TBAC) is an authorization management model [36] that combines type-based access control with instance and usage based access control. In TBAC, a type-based access control module is used to make the authorization decision, and the authorization life-cycle (activation and deactivation) is controlled by the instance and usage based access control module. The focus of TBAC is to manage the workflow of the authorization process so there is no specific authorization mechanism proposed for making access authorization decisions. DRLA is an authorization protocol, which provides exactly what TBAC neglects. Thus, this emerging protocol could potentially

offer a new area of research, focusing on the benefits of integrating TBAC and DRLA models.

5.5. Summary

In this Chapter, we presented the simulation results and analyzed new functions provided in DRLA. We further compared the functional differences between DRLA and other approaches. Based on the simulation results, we confirmed that DRLA addresses the missing functions in other models.

Chapter 6: CONCLUSION AND FUTURE WORK

A distributed computing system, such as a Grid, could be a very dynamic environment which includes user groups, network topologies, *etc.* A user group could be formed by the users from different networks, organizations, and administrative domains with different hardware and software infrastructures and/or managerial policies. When handling requests from a wide range of users from different domains, it becomes a challenge to accommodate all the differences. It is impossible for each service provider to track all users (the number of users could potentially be very large) in a Grid. Therefore, an access control mechanism that provides a user with appropriate access to the resources in a dynamic environment is required. RBAC models have been demonstrated to be an effective and efficient approach for an administrator to manage access to a computing system. Much has been done to adapt the RBAC concept to Grids, focusing on verification of the dynamic factors and contexts of a user, such as time, location, and rank. Some applications also allow administrators to change the policies during the authorization process. However, the reviewed literature has revealed no implementation that handles the real-time and on-demand authorization with consideration of the dynamic factors in a Grid. Therefore, the absence of a mechanism that facilitates resource sharing in a dynamic computing environment necessitated the development of a new dynamic authorization protocol, Dynamic Role Lease Authorization (DRLA) [121], which is suitable for a dynamic distributed computing environment.

6.1. Contributions

The proposed protocol, DRLA, [121] provides a mechanism for dynamic authorization in a distributed computing system, such as a Grid, which is highly heterogeneous and dynamic. As discussed in Section 3.3, the role leasing structure is introduced in DRLA to utilize the resources in a Grid. In such an environment, some idle resources could be assigned to a non-active or disconnected users due to some dynamic factors or failures. The role leasing structure leases out a role to use a resource; the resource is released back to the system once the lease is expired or not renewed. This approach limits a resource so that it can only be occupied within the leasing period, implying that any failure may only delay the resource until the completion of the leasing period. On the one hand, the leasing structure minimizes the impact caused by a failure. On the other hand, it allows a short absence or disconnection from the system for some dynamic situations. For example, a user may require a short disconnection to switch from one access point to another. The user can maintain the access as long as the connection is re-established within the leasing period and all policies are still satisfied.

In Section 3.1, we introduced how DRLA maintains the minimum privileges for each user, and dynamically assigns and grants the required privilege to the corresponding authorized user. All other accesses for performing other non-essential tasks are granted on demand. There are two main dynamic aspects in the proposed protocol. First, the user access is authorized based on not only the user's role but also the user's contextual information, such as requested time and location. The protocol also handles the policy changes that may be made by the resource providers in the system. All the contextual information is verified for each request to ensure that each request satisfies the

authorization criteria that is specified in the policies. The dynamic contexts and (business and database) policies are verified before the access is granted; this approach minimizes the security risk that has been identified in the “turn on/off” approach, which has been used in many current access control systems. Achieving these two aspects is essential in a Grid access control system. The second dynamic characteristic in DRLA is the integrated leasing concept, which minimizes the security threads that may be caused by long holding permissions. With the leasing approach, all privileges are associated with a lease that specifies the time limit and any associated conditions of the lease. All the privileges will be revoked once the lease has expired. In order to retain the granted privilege, the user must renew the lease before its expiry. The dynamic characteristics embedded in DRLA protocol enable an access control system to minimize the security concerns that are caused by the dynamic factors in a Grid environment. The simulation that is performed in this thesis illustrated the feasibility of implementing the protocol, as well as the stable performance in a simulated distributed environment.

As discussed in Section 5.4, the authorization solution provided by the DRLA protocol has overcome a significant limitation of many dynamic or active authorization models. In summary, many existing models have recognized the importance of user contextual parameters and made use of those parameters in the authorization processes. However, many models have missed the dynamic nature of the contextual information in a heterogeneous and dynamic environment, such as a Grid. DRLA introduces the role leasing concept to address this problem and provides the features that are described in this thesis.

In Section 3.3.ii, we presented the DRLA on-going dynamic context aware authorization mechanism for a Grid environment. The authorization process in a highly dynamic computing system should not be a one-time process; it should be an on-going procedure to ensure that users satisfy the authorization requirements all the time. Unlike many other authorization models, DRLA establishes an effective and straightforward on-going mechanism by combining the Lease and RBAC concepts to monitor users' contextual information and enforce the security requirements.

6.2. Research Direction

The proposed protocol has been evaluated in a simulated environment; however, it is important to extend the study to a real Grid or Cloud environment, such as WestGrid. This continued real-world study should refine the protocol to cover some practical problems in such settings, such as combining different network topologies. In addition, this research could be extended to include incorporation with other grid components, for example the scheduling system and policy management system that were mentioned earlier. Although these topics are out the scope of this thesis, they are critical for a successful Grid implementation. Therefore, a deeper study of the relationship between the dynamic nature of these components and an access control system should be performed.

A Grid could be formed from multiple sub-Grids/Clouds. Currently, due to the heterogeneity and dynamic factors in all VOs there is no Grid that can group all sub-Grids together and form a federated Grid (the Grid) - a scientific vision for much research. To achieve this, there are many challenges that need to be overcome, such as security, knowledge mining, information integration, and network infrastructure. To

provide a commonly trusted environment, access control becomes a critical component in such an environment – providing the right information and resources for the right people at the right time and space, as needed. Some main characteristics of the Grid environment would be one that is highly heterogeneous, highly dynamic, highly decentralized, and so on.

With the analysis done in Section 5.4, we have identified some authorization models that focus on different aspects in the authorization process. For example, Wang [37] has proposed a model to address policy management and Thomas *et al.* [36] have proposed an authorization workflow management model. DRLA has recognized the benefits provided by these models and has realized the opportunity to utilize these models in providing an enhanced authorization service.

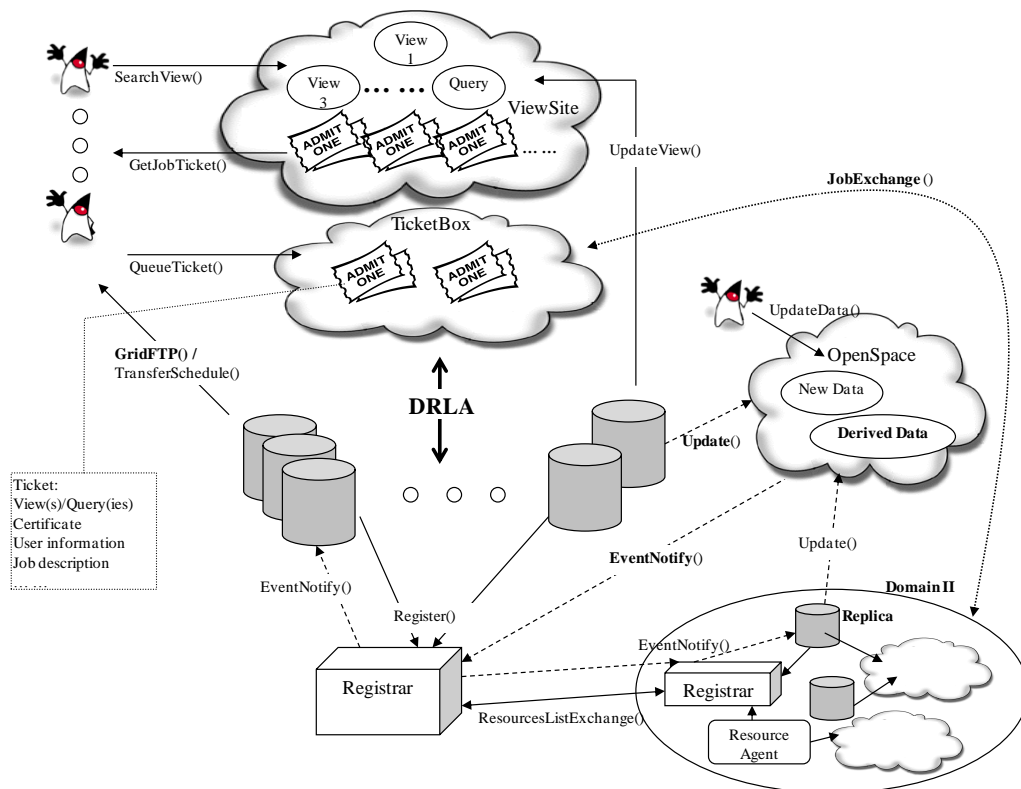


Figure 6-1: Grid Data Access Management with DRLA and Tuple Spaces

For an access control system in such an environment, overcoming these characteristics should be one of the main focuses. To address that, there are some research studies/projects proposed that apply the Tuple Space concept for building the Grid. We believe that Tuple Space could be one of the effective ways to build the Grid, and DRLA could work well in a Tuple Space based Grid. To accomplish that, there are multiple areas that we need to study further, for example, managing the leases and implementing the GIS service in a Tuple Space environment.

The conceptual model that is shown in Figure 6-1 proposes a new data access management system for the Grid using the Tuple Spaces paradigm and DRLA protocol proposed in this thesis. In this model, there are four major user cases in the Grid environment: User, DB or Resource Agent, Registrar, and Tuple Spaces (ViewSite and TicketBox). Users check the views (data resources) that are available in the Grid. Users can get the ticket and submit a query job to the TicketBox. Before a job can be executed, it has to be authorized through the DRLA processes. Job submission does not require any knowledge, such as the location, network domain, ownership, and operational procedure, of the resources. Users simply submit a query into the space and wait for the result from the DB agent. The information regarding the resource's availability is published into the Space by the Registrar. The Tuple Spaces are the center point and act as a communication buffer in this system [38, 39]. There are three types of tuples that will exist: View, Ticket, and Resources List. The Space manages the jobs submitted by Grid users and waits for the available resource to claim the jobs. In addition, it maintains the leasing status of each job. If the result of a claimed job does not return back to the space, or the lease has not been renewed within the leasing time limit, then the job will be released back to the space.

The leasing mechanism allows the system to provide fault tolerance services. A job result will be kept in the space for the corresponding user to claim. The Resources List is posted and updated by the Registrar. This List provides the information of available resources in the network domain. This data access management system inherits the benefit from the Tuple Space framework and DRLA protocol for a Grid environment. Information (or data) is the most important asset in the computing world and human decision-making process, and the Grid computing concept aims high allowing everyone to share the asset for their needs. The Tuple Space framework and DRLA protocol provides the new data access management system a flexible and secured environment to share the data; this research brings us one step forward to the reality of “The Grid”

Bibliography

1. R. S. Sandhu, *et al.*, "Role-Based Access Control Models", IEEE Computer 29(2): 38-47, IEEE Press, 1996.
2. David Ferraiolo and Richard Kuhn. "Role-based access controls". In 15th NIST-NCSC National Computer Security Conference, pages 554-563, Baltimore, MD, Oct 1992.
3. Peter Burkholder, "SSL Man-in-the-Middle Attacks", February, 2002, <http://www.pburkholder.com/sysadmin/SSL-mitm/index.php>
4. Virtual Private Network Consortium, "VPN Technologies: Definitions and Requirements", White Paper, <http://www.vpnc.org/vpn-technologies.html>
5. David F. Ferraiolo, D. Richard Kuhn, Ramaswamy Chandramouli, "Role-Based Access Control", 2003, ISBN: 1580533701
6. Junzhe Hu and Alfred C. Weaver, "A Dynamic, Context-Aware Security Infrastructure for Distributed Healthcare Applications", Pervasive Security, Privacy and Trust (PSPT2004), Boston, MA, August 2004
7. Hongxia Cai; Tao Yu; Minglun Fang, "Dynamic access control for manufacturing grid," Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on , vol.2, no., pp. 917-920, 12-14 Oct. 2005
8. 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.
9. Yao Hanbing , Hu Heping, Lu Zhengding and Li Ruixuan, "Dynamically authorized role-based access control for grid applications", Geo-Spatial Information Science Journal, Issue Volume 9, Number 3 / September, 2006, Pages 223-228
10. Hong Fan, Zhu Xian and Xing Guang-lin, "A distributed role-based access control model for multi-domain environments", Journal of Shanghai University, Issue Volume 10, Number 2 / April, 2006, Pages 134-141
11. CHEN Jian-gang, WANG Ru-chuan, WANG Hai-yan, "The extended RBAC model based on grid computing", The Journal of China Universities of Posts and Telecommunications, Issue 13 volume 3, 2006, Pages 93-97
12. Hai Jin, Weizhong Qiang, Xuanhua Shi, and Deqing Zou, "RB-GACA: A RBAC based Grid Access Control Architecture", International Journal of Grid and Utility Computing, Inderscience Enterprises Ltd., Vol.1, No.1, 2005, pp.61-70.
13. A.L. Pereira; V. Muppavarapu; S.M. Chung, "Role-based access control for grid database services using the community authorization service," Dependable and Secure Computing, IEEE Transactions on , vol.3, no.2, pp.156-166, April-June 2006
14. Schneier, Bruce. "Secrets and Lies : Digital Security in a Networked World". John Wiley & Sons, 2000, ISBN: 978-0471253112
15. Phoomvuthisarn, S., "Trust and role based access control for secure interoperation ("TracSI")," Communications and Information Technologies, 2007. ISCIT '07. International Symposium on , vol., no., pp.1458-1463, 17-19 Oct. 2007
16. Chandramouli, R., "A framework for multiple authorization types in a healthcare application system", Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual , vol., no., pp. 137-148, 10-14 Dec. 2001
17. Manoj Sastry, Ram Krishnan and Ravi Sandhu, "A New Modeling Paradigm for Dynamic Authorization in Multi-domain Systems", MMM-ACNS 2007.
18. Demchenko, Y.; de Laat, C.; Gommans, L.; van Buuren, R., "Domain Based Access Control Model for Distributed Collaborative Applications," e-Science and Grid Computing, 2006. e-Science '06. 2nd IEEE Intl Conference on, pp.24-24, Dec. 2006
19. Jin Wu; Leangsuksun, C.B.; Rampure, V.; Hong Ong, "Policy-Based Access Control Framework for Grid Computing," Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on , vol.1

20. J. Barkley, K. Beznosov, and J. Uppal, "Supporting Relationships in Access Control Using Role Based Access Control," In Proceedings of ACM Role-based Access Control Workshop, Fairfax, Virginia, USA, 1999, pp. 55-65.
21. Ellison, C. and B. Schneier. "Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure," Computer Security Journal, v 16, n 1, 2000, pp. 1-7
22. Yao, H., Hu, H., Huang, B., and Li, R., "Dynamic Role and Context-Based Access Control for Grid Applications", PDCAT 2005. IEEE Computer Society, p. 404-406.
23. Ying Chen, Shoubao Yang, Leitao Guo, "Dynamic-Role Based Access Control Framework Across Multi-domains in Grid Environment", GCC 2005: 161-165
24. Oh, S. and Park, S. "Task-Role Based Access Control (T-RBAC): An Improved Access Control Model for Enterprise Environment". Lecture Notes in Computer Science, vol. 1873/2000. Springer-Verlag, London, 264-273.
25. Hong Fan, Zhu Xian and Xing Guanglin, "Distributed role-based access control for coaliation application", Journal Geo-Spatial Information Science, Issue Volume 8, Number 2 / June, 2005, Pages 138-143
26. Yu Guangcan , Li Zhengding , Li Ruixuan and Mudar Sarem, "Centralized role-based access control for federated multi-domain environments", Wuhan University Journal of Natural Sciences, Issue Volume 11, Number 6 / November, 2006, p. 1688-1692
27. S. Cannon, S. Chan, D. Olson, C. Tull, V. Welch, L. Pearlman, "Using CAS to Manage Role-Based VO Sub-Groups", CHEP03, Mar., 2003, La Jolla, California
28. Jongil Jeong, Weehyuk Yu, Dongkyoo Shin, Dongil Shin, Kiyoun Moon, Jaeseung Lee, "Integration of Single Sign-On and Role-Based Access Control Profiles for Grid Computing", APWeb 2006: 880-885
29. Yingjie Xia; Guanghua Song; Yao Zheng; Mingzhe Zhu, "R2P: A Peer-to-Peer Transfer System Based on Role and Reputation," Knowledge Discovery and Data Mining, 2008. WKDD 2008. International Workshop on , pp.136-141, Jan. 2008
30. Saltzer, Jerome H. & Schroeder, Michael D. "The Protection of Information in Computer Systems," 1278-1308. Proceedings of the IEEE 63, 9 (September 1975).
31. David Wheeler, "Secure programmer: Minimizing privileges", available at <http://www.ibm.com/developerworks/linux/library/l-sppriv.html>
32. Yuqing Sun; Bin Gong; Xiangxu Meng; Zongkai Lin, "Active Authorization Management for Multi-domain Cooperation," Computer Supported Cooperative Work in Design, 2007. pp.162-167, 26-28 April 2007
33. Adaikkalavan, R.; Chakravarthy, S., "Active Authorization Rules for Enforcing Role-Based Access Control and its Extensions," Data Engineering Workshops, 2005. pp. 1197-1197, April 2005
34. Mattas, A.K.; Mavridis, I.K.; Pangalos, G.I., "Towards dynamically administered role-based access control," Database and Expert Systems Applications, 2003. pp. 494-498, 1-5 Sept. 2003
35. Mattas, A.K.; Mavridis, I.K.; Pangalos, G.I., "A Paradigm for Dynamic and Decentralized Administration of Access Control in Workflow Applications," IFIP International Federation for Information, Volume 201/2006, pp. 196-207, July 26, 2006
36. Thomas, R. K. and Sandhu, R. S. "Task-Based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-Oriented Authorization Management", IFIP Tc11 Wg11.3 Eleventh international Conference on Database Security Xi: Status and Prospects 1997
37. Wang, C., "Dynamic Access Control Prediction for Ordered Service Sequence in Grid Environment". IEEE/WIC/ACM international Conference on Web intelligence, 2006
38. Nelson C.N. Chu, Ken E. Barker, "Grid Scheduling Hub," SKG, 2008 Fourth International Conference on Semantics, Knowledge and Grid
39. Nelson C.N. Chu, Quang M. Trinh, Ken E. Barker, and Reda S. Alhadjj, "A Dynamic Ontology Mapping Architecture for a Grid Database System," SKG, 2008 Fourth International Conference on Semantics, Knowledge and Grid
40. Sturgis, H., Mitchell, J., and Israel, J., "Issues in the design and use of a distributed file system", SIGOPS Oper. Syst. Rev. 14, 3 (Jul. 1980), 55-69
41. Saltzer, J. H. and M. D. Schroeder. "The protection of information in computer systems", IEEE vol. 63, no. 9, 1975, pp. 1278-1308

42. P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. "The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments". National Information Systems Security Conference, pp. 303–14, Oct. 1998
43. Zhao Song-zheng; Xu Heng; Gao Na; Sun Yi-ran, "Dynamic Authorization Management Model Based on ASP.NET and WBS", Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference, Page(s):6258 - 6261
44. Xiao Ying, "Security in distributed, grid, mobile, and pervasive computing", 2007, Auerbach Publications, ISBN-10: 0-8493-7921-0
45. Cary G. Gray, David R. Cheriton, "Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency", in Proc. 12th Symp on Operating Systems Principles, Dec 1989
46. W. Keith Edwards, "Core Jini 2nd Edition", Prentice Hall, ISBN 0-13-089408-7
47. Fleisch, B. and Popek, G. "Mirage: a coherent distributed shared memory design". ACM Symposium on Operating Systems Principles SOSP '89, 211-223
48. M.L. Kazar, "Synchronization and Caching Issues in the Andrew File System" Proc. Usenix Winter Tech. Conf., Usenix Assoc., Berkeley, Calif., Feb. 1988
49. Howard, J. H., Kazar, M. L., Menees, S. G., Nichols, D. A., Satyanarayanan, M., Sidebotham, R. N., and West, M. J. "Scale and performance in a distributed file system". ACM Trans. Comput. Syst. 6, 1 (Feb. 1988), 51-81.
50. Nelson, M. N., Welch, B. B., and Ousterhout, J. K. 1988. "Caching in the Sprite network file system", ACM Trans. Comput. Syst. 6, 1 (Feb. 1988), 134-154.
51. Khadilkar, M., Feamster, N., Sanders, M., and Clark, R. 2007. "Usage-based dhcp lease time optimization", ACM SIGCOMM Conference on internet Measurement, IMC '07, 71-76.
52. Jeanne Lim, "Study: Grid adoption growing fastest in AP", <http://www.zdnet.com/study-grid-adoption-growing-fastest-in-ap-2039379387/>
53. <http://www.unt.edu/cedr/gridcomputing.pdf>
54. Matt Haynos, "Perspectives on grid: Grid computing -- next-generation distributed computing", <http://ftp.utcluj.ro/pub/docs/cursuri/tarc/GRID/gr-heritage.pdf>
55. Paul Yao, David Durant, "Programming .NET Compact Framework 3.5", 2nd Edition, Addison-Wesley Professional, ISBN-10: 0-321-57358-7
56. Andrew Troelsen, "Pro C# with .NET 3.0", Special Edition, Apress, ISBN-10: 1590598237
57. Scott McLean; James Naftel; Kim Williams, "Microsoft .NET Remoting", Microsoft Press, ISBN-10: 0-7356-1778-3
58. Sun Microsystems, Inc., "TCP/IP and Data Communications Administration Guide", Part No: 805-4003-10, October 1998
59. Charles Kozierok, "The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference", No Starch Press, 2005, ISBN-10: 159327047X
60. Butler Lampson, "Designing a global name service", ACM Symposium on Principles of Distributed Computing, Minaki, Ontario, 1986, pp 1-10
61. Bill Venners, "Jini: New technology for a networked world", <http://www.javaworld.com/jw-06-1999/jw-06-jiniology.html>
62. R. Droms, "Dynamic Host Configuration Protocol, RFC-2131", Network Working Group, IETF, <ftp://ftp.isi.edu/in-notes/rfc2131.txt>
63. Borja Sotomayor, Lisa Childers, Morgan Kaufmann, "Globus Toolkit 4: Programming Java Services", First Edition, Morgan Kaufmann, ISBN: 0123694043
64. M. Burrows, "Efficient Data Sharing", Ph.D. Thesis, Cambridge University Computer Laboratory, September 1988. Also available as Technical Report No. 153, December 1988.
65. Mann, Timothy and Hisgen, Andy and Swart, Garret, "An algorithm for data replication", Digital Systems Research Center Tech. Report, 1989
66. Satyanarayanan, M., Howard, J. H., Nichols, D. A., Sidebotham, R. N., Spector, A. Z., and West, M. J., "The ITC distributed file system: principles and design", ACM Symposium on Operating Systems Principles, SOSP '85, 35-50.
67. Mockapetris, P. V. 1987, "Domain Names - Concepts and Facilities", RFC-1035, IETF
68. Allman, M., Paxson, V., and Stevens, W. 1999, "TCP Congestion Control", RFC-2581, IETF
69. Barr, D. 1996, "Common DNS Operational and Configuration Errors", RFC-1912, IETF
70. Detlefs, D., Flood, C., Heller, S., and Printezis, T. 2004. "Garbage-first garbage collection", International Symposium on Memory Management ISMM '04. ACM, 37-48.

71. David W. Wall., "Mechanisms for BToudcast and Selective Broadcast", PhD thesis, Stanford University, California, U.S.A., June, 1980
72. Larson, Per-Åke (April 1988), "Dynamic Hash Tables", Communications of the ACM 31: 446–457, doi:10.1145/42404.42410
73. Hubert Zimmermann, "OSI Reference Model", IEEE Transactions on Communications, Vol. COMM-28(4), p425, April 1980
74. P. Chown 2002, "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security", (TLS), RFC-3268, IETF
75. Tom Olzak, "DNS Cache Poisoning: Definition and Prevention" , March 2006, http://adventuresinsecurity.com/Papers/DNS_Cache_Poisoning.pdf
76. Rüdiger Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications", International Conference on Peer-to-Peer Computing, IEEE 2001.
77. G. Camarillo, Ed., "Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability", RFC-5694, November 2009
78. Rajkumar Buyya and Manzur Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", Concurrency and Computation: Practice and Experience, 2002
79. Qun Ni, Shouhuai Xu, Elisa Bertino, Ravi Sandhu, and Weili Han. "An Access Control Language for a General Provenance Model", VLDB Workshop on Secure Data Management (SDM '09)
80. E. Stokes, D. Byrne, B. Blakley, and P. Behera, RFC-2820, "Access Control Requirements for LDAP", May 2000
81. I. Foster, Y. Zhao, I. Raicu, S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, 2008. GCE '08 , vol., no., pp.1-10, 12-16 Nov. 2008
82. Kate Craig-Wood, "Definition of Cloud Computing, incorporating NIST and G-Cloud views", <http://www.katescomment.com/definition-of-cloud-computing-nist-g-cloud/>
83. Grance, T., Mell, P. M., "The NIST Definition of Cloud Computing", 2011, The National Institute of Standards and Technology (NIST)
84. María S. Pérez, "Grid and Cloud Computing" , Universidad Politécnica de Madrid, http://laurel.datsi.fi.upm.es/_media/docencia/asignaturas/ccg/gridcloud.pdf
85. Jansen Wayne, Grance Timothy, "Guidelines on. Security and Privacy in Public Cloud Computing", 2011, The National Institute of Standards and Technology (NIST)
86. William Voorsluys, J Broberg, R Buyya, "Introduction to Cloud Computing", 2011, John Wiley & Sons, Inc., Hoboken, New Jersey
87. Judith Myerson, "Cloud computing versus grid computing", <http://www.ibm.com/developerworks/web/library/wa-cloudgrid/>, 03 Mar 2009
88. Preeti Sunil, "Cloud Computing Vs. Grid Computing", 1/30/2012, <http://www.buzzle.com/articles/cloud-computing-vs-grid-computing-how-do-they-differ.html>
89. Jensen, M.; Schwenk, J.; Bohli, J.; Gruschka, N.; Iacono, L.L., "Security Prospects through Cloud Computing by Adopting Multiple Clouds", Cloud Computing (CLOUD), 2011 IEEE International Conference
90. Jianxin Li; Bo Li; Zongxia Du; Linlin Meng, "CloudVO: Building a Secure Virtual Organization for Multiple Clouds Collaboration", Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 2010 ACIS International Conference
91. Bernstein, D.; Vij, D., "Intercloud Security Considerations", Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference
92. AlZain, M.A.; Pardede, E.; Soh, B.; Thom, J.A., "Cloud Computing Security: From Single to Multi-clouds", System Science (HICSS), 2012 Hawaii International Conference
93. Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., and Morrow, M., "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability". In Proceedings of ICIW '09, the Fourth International Conference on Internet and Web Applications and Services, pp. 328-336 (2009).
94. Kuyoro S. O., Ibikunle F. & Awodele O, "Cloud Computing Security Issues and Challenges", MIPRO, 2010 International Convention

95. Rohit Bhadauria, Rohit Bhadauria, "A Survey on Security Issues in Cloud Computing", 2011, IEEE Communications Surveys and Tutorials
96. Almorsy, M., "Collaboration-Based Cloud Computing Security Management Framework", Cloud Computing (CLOUD), 2011
97. Pippal, S.; Sharma, V.; Mishra, S.; Kushwaha, "An Efficient Schema Shared Approach for Cloud Based Multitenant Database with Authentication and Authorization Framework ",D.S., P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011
98. Liu, Wentao, "Research on cloud computing security problem and strategy", Consumer Electronics, Communications and Networks (CECNet), 2012
99. S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", Journal of Network and Computer Applications, 34(1), 2011, pp 1-11.
100. Shuai Zhang; Xuebin Chen; Shufen Zhang; Xiuzhen Huo, "The comparison between cloud computing and grid computing ",Computer Application and System Modeling (ICCASM), 2010
101. Rak, M.; Liccardo, L.; Aversa, R., "A SLA-based interface for security management in cloud and GRID integrations", Information Assurance and Security (IAS), 2011
102. Almulla, S.A.; Chan Yeob Yeun, "Cloud computing security management", Engineering Systems Management and Its Applications (ICESMA), 2010
103. María S. Pérez, "Grid and Cloud Computing", Facultad de Informática, Universidad Politécnica de Madrid
104. Pereira, A.L., "RBAC for High Performance Computing Systems Integration in Grid Computing and Cloud Computing ",Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011
105. James B. D. Joshi , Elisa Bertino , Usman Latif , Arif Ghafoor, "TRBAC: A Temporal Role -based Access Control Model", ACM Transactions on Information and System Security (TISSEC) 2001
106. Rashid, Z.; Basit, A.; Anwar, Z., "TRDBAC: Temporal reflective database access control", Emerging Technologies (ICET), 2010
107. Zhenghong Liu; Yuhong Jia; Tieli Sun, "Study on Role-Based Access Control with Time Constraint", Computational Intelligence and Security (CIS), 2011
108. R. Bhatti, B. Shafiq, E. Bertino, and A. Ghafoor, "X-GTRBAC Admin: A Decentralized Administration Model for Enterprise-Wide Access Control", ACM Trans. on Information and System Security, vol. 8, no. 4, pp. 388-423, 2005.
109. H. Jin, W. Qiang, X. Shi, and D. Zou, "RB-GACA: A RBAC Based Grid Access Control Architecture", Int'l Journal of Grid and Utility Computing, vol. 1, no. 1, pp. 61-70, 2005.
110. A. L. Pereira, C. W. Moseley, D. L. Ferron, B. A. VanTreeese, D. L. Goree and K. M. Kirch, "A Framework for Semantic-Based Dynamic Access Control in Data Grids", Proc. 10th Linux Cluster Institute (LCI) Int'l Conf. on High Performance Clustered Computing, 2009.
111. OpenStack, <http://openstack.org/>
112. Globus, <http://www.globus.org/>
113. Uwe Schwiegelshohn, Rosa M. Badia, Marian Bubak, Marco Danelutto, Schahram Dustdar, Fabrizio Gagliardi, Alfred Geiger, Ladislav Hluchy, Dieter Kranzlmler, Erwin Laure, Thierry Priol, Alexander Reinefeld, Michael Resch, Andreas Reuter, Otto Rienhoff, Thomas Ruter, Peter Sloat, Domenico Talia, Klaus Ullmann, Ramin Yahyapour, and Gabriele von Voigt. 2010. Perspectives on grid computing. Future Gener. Comput. Syst. 26, 8 (October 2010), 1104-1115. <http://dx.doi.org/10.1016/j.future.2010.05.010>
114. Anthony Sulistio and Rajkumar Buyya, A Grid Simulation Infrastructure Supporting Advance Reservation, Proc. of the 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), pp. 1-7.
115. ALEA 3, <http://www.fi.muni.cz/~xklusac/alea/>
116. Efeng Lu, Zhihong Xu, and Jizhou Sun, An Extendable Grid Simulation Environment Based on GridSim, Grid and Cooperative Computing - Lecture Notes in Computer ScienceVolume 3032, 2004, pp 205-208
117. Deelman, E., Singh, G. ; Livny, M. ; Berriman, B. ; Good, J., The Cost of Doing Science on the Cloud: The Montage Example, High Performance Computing, Networking, Storage and Analysis, 2008.

118. Archit Kulshrestha, Quality Of Service Based Data-Aware Scheduling, PhD Dissertation, Louisiana State University and Agricultural and Mechanical College, May 2011, <http://etd.lsu.edu/docs/available/etd-04202011-091322/unrestricted/kulshresthadiss.pdf>
119. Alexandru Iosup, A Framework for the Study of Grid Inter-Operation Mechanisms, PhD Dissertation, 2008, Delft University of Technology, http://www.pds.ewi.tudelft.nl/pubs/ph_d/iosup.pdf
120. M. Krystek, K. Kurowski, A. Oleksiak, and W. Piateket., Energy-aware simulations with GSSIM, COST Action IC0804 on Energy Efficiency in Large Scale Distributed Systems
121. Nelson Chu and Ken Barker, "Dynamic Role Lease Authorization for a Grid/Cloud", IEEE ICT and Knowledge Engineering Conference, Bangkok, Thailand, November 2012