Security for Mobile Grid Systems

Tariq Falah Alwada'n

Software Technology Research Laboratory

Faculty of Technology

De Montfort University

United Kingdom, England

A thesis submitted for the degree of Doctor of Philosophy

2012

Abstract

Grid computing technology is used as inexpensive systems to gather and utilize computational capability. This technology enhances applications services by arranging machines and distributed resources in a single huge computational entity. A Grid is a system that has the ability to organize resources which are not under the subject of centralized domain, utilize protocols and interfaces, and supply high quality of service. The Grid should have the ability to enhance not only the systems performance and job throughput of the applications participated but also increase the utilization scale of resources by employing effective resource management methods to the huge amount of its resources. Grid mobility appears as a technology to facilitate the accomplishment of requirements for Grid jobs as well as Grid users. This idea depends on migrating or relocating jobs, data and application software among Grid nodes. However, making use of mobility technology leads to data confidentiality problems within the Grid. Data confidentiality is the protection of data from intruders' attacks. The data confidentiality can be addressed by limiting the mobility to trusted parts of the Grid, but this solution leads to the notion of Virtual Organizations (VOs). Also as a result of mobility technology the need for a tool to organize and enforce policies while applying the mobility has been increased. To date, not enough attention has been paid to policies that deal with data movements within the Grid. Most existing Grid systems have support only limited types of policies (e.g. CPU resources). A few designs consider enforcing data policies in their architecture. Therefore, we propose a policy-managed Grid environment that addresses these issues (user-submitted policy, data policy, and multiple VOs).

In this research, a new policy management tool has been introduced to solve the mobility limitation and data confidentiality especially in the case of mobile sharing and data movements within the Grid. We present a dynamic and heterogeneous policy management framework that can give a clear policy definition about the ability to move jobs, data and application software from nodes to nodes during jobs' execution in the Grid environment. This framework supports a multi-organization environment with different domains, supports the external Grid user preferences along with enforces policies for data movements and the mobility feature within different domains.

The results of our research have been evaluated using Jade simulator, which is a software framework fully implemented in Java language and allows agents to execute tasks defined according to the agent policy. The simulation results have verified that the research aims enhance the security and performance in the Grid environments. They also show enhanced control over data and services distribution and usage and present practical evidence in the form of scenario test-bed data as to the effectiveness of our architecture.

Acknowledgements

I would first like to thank my God, without whom none of this work would have been possible.

My sincere thanks and deep gratitude go to my supervisor Dr Helge Janicke who provides me with supervision, guidance, research input, and patience in the completion of this research. It has been a pleasure to have the opportunity to work with you all during the course of this research. Also I would like to thank my second supervisor Dr Omer Aldabbas for his advice, encouragement and guidance on many aspects relating to my research. He constantly encouraged me to remain open-minded and always to explore beyond accepted boundaries.

My father's inexhaustible store of knowledge and experience proved invaluable during my studies, and he imparted it as to a friend and equal rather than as to a son. His critical approach to the world inspired in me a similar attitude which has served me well, in my life as for this study. My mother's loving care saw me through many difficult times as this work progressed. I thank them both from the bottom of my heart.

I am thankful to my brothers Mohammed and Firas as well as my sisters Tagreed, for their unfailing encouragement and support in everything I have tried to do throughout my life. They have been a tower of strength without whom I could not have achieved what I have done, and who have made me the person I am.

I wish to express my thanks to all my colleagues in the STRL Group for their support, especially Hamza Aldabbas for his help and support. I am exceedingly fortunate in having such warm and generous friends in Jordan (Abd Alsalam, Thair, Belal, Fawaz, Hazem and their families) and in the UK (Naser, Ahmed, Mhmoud, Mo-

hammed Alhelali and Mazen Barkat). Because of the pressures of my study, I proved a very bad correspondent, but they did not give up on me. To them I say, “Thank you for your persistence in keeping in touch with me when I did not reciprocate on so many occasions. Your perseverance is hugely appreciated.”

Also I would like to say thanks to my best friends Anne and Mohammed for the outstanding job they did on proofreading my thesis. Thank you so much for your diligence in getting my thesis edited in such a short time.

Finally, I would like to deeply thank my cousin Mohammed Alwadan for his support and guidance through my master and PhD degree and all of my uncles and aunts for their supports.

Declaration

The thesis presented here is my own and original. It is submitted for the degree of Doctor of Philosophy at the Software Technology Research Laboratory (STRL), Faculty of Technology, De Montfort University, United Kingdom.

Publications

1. Tariq Alwada'n, Hamza Aldabbas, Helge Janicke, Thair Khdour and Omar Aldabbas. "Dynamic Policy Management in Mobile Grid Environments". International Journal of Computer Networks and Communications (IJCNC), Vol. 4, No. 2, March 2012 .

2. Tariq Alwada'n, Helge Janicke, Omer Aldabbas and Mai Alfawair. "New Framework for Policy Support for Mobile Grid Services". Risk and Security of Internet and Systems (CRiSIS), IEEE Computer Society. Timisoara, Romania 2011.

3. Tariq Alwada'n, Helge Janicke, Omer Aldabbas and Hamza Aldabbas. "New Framework for Dynamic Policy Management in Grid Environments". In Proceedings of CoNeCo2011 in Communications in Computer and Information Science, Volume 162, Part 2, pp. 297-304, Ankara, Turkey (LNCS, Springer) 2011.

4. Hamza Aldabbas, Tariq Alwada'n, Helge Janicke and Ali Al-Bayatti. "Data Confidentiality in Mobile Ad hoc Networks". International Journal of Wireless and Mobile Networks (IJWMN) Vol. 4, No. 1, February 2012.

5. Hamza Aldabbas, Helge Janicke , Radwan AbuJassar and Tariq Alwada'n. "Ensuring Data Confidentiality And Privacy In Mobile Ad hoc Networks". The 3rd International Conference on Wireless and Mobile Networks (WiMoNe-2011), (LNICST, Springer),Bangalore, India, January 2012.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the advances in communication technology and global system of interconnected computer networks (internet), Grid computing appear as a result of a combination of multi-network computer system to develop a wide range and heterogeneous system used to solve scientific or industrial problems [50]. A grid is a system that should have the ability to organize resources (resources here refer to the management of computing resources. For example: computer, software applications, etc.) which are not subject to a centralized domain, utilize protocols and interfaces and supply high quality of service [58]. Thus, the major advantage of Grid computing is the capability to organize and share resources [25], [90]. As a result of such technology many challenges need to be overcome to develop this technology, such as finding suitable resources and reducing the number of rejected jobs. There are a lot of contributions to solve some of these challenges such as grid mobility.

Mobility is the ability to migrate or relocate jobs, data and application software among grid nodes. These migrations depend on the grid's users and the grid's nodes policies. Mobility facilitates the accomplishment of requirements for grid jobs as well as grid users. It also assists grid evolution, improves performance of operating applications by relocating data to the target host, therefore reducing the communication consumption and reducing load balancing issues. David G. Rosado et.al [117] described the mobility as "In the purview of Grid and Mobile Computing, Mobile Grid is an heir of the Grid,

which addresses mobility issues, with the added elements of supporting mobile users and resources in a seamless, transparent, secure and efficient way [112], [116], [17]". Computational mobility may also be known as a control migration, data migration, link and object migration [28]. This type of migration allows the data and codes to migrate and execute on various systems across the network. Also it offers movable execution control and the ability to connect software elements at runtime whilst migrating from one system to another and back to the original system again. Sze-Wing Wong et.al [63] have introduced a new mobile grid services that can enhance the fixed grid service with migration capability, but they did not give enough attention to the policy aspects of their design.

Policies are groups of regulations, standards and practices written by the administrators of resources about how their resources or jobs can be handled and used. Every resource applies its own security policy that may result in the refusal of requests for utilizing its resources. Security has become a critical aspect in checking the subject trying to use a service (authentication), and in verifying whether it is allowed or not, to use the service (authorization). Policies specify the way that a specific job should be accomplished, how security is applied in a domain and how an organization organizes, secures and distributes their resources.

1.1 Research Motivation

Any secure grid environment should provide mechanisms to secure authentication, resource protection, authorization, communication, data transfer and encryption [77]. One of the most important security challenges that face the grid environment is coordinating users' identities among local and wide networks and dealing with the variety of local security techniques and trust relationships between resources and users.

It is important to be able to locate and detect the available resources within the grid environment, and to map jobs to these resources later with respect to the policies of each entities. The problem begins when policy checking for the resources and jobs takes place in computational grids. The heterogeneity of policies and attributes leads

to a need for policy management tools which can handle both diversity and heterogeneity in these policies. In the Globus Toolkit [13], before the job submission, there are many steps for authenticating users who request resources [45],[118]. However, after the authentication, there are no further resource access restrictions on how to use the resources. This is known as "all or nothing".

Before the users can submit their jobs or run their applications on a certain source or system they need to assert that this source or system has not been compromised which could result in their own application or data being stolen.

Currently, there has been much research that focuses on policy management in the Grid environment [29],[75],[80],[113],[115]. The aim of policy management is "to apply an integrated management system so that system management, network management, and application management can co-operate in grid computing" [119].

Each job has different requirements and specifications in order to be executed in the grid. The grid is a multi-organization environment with different institutes. Each institute might want to apply some boundaries on how its resources are being utilized by other institutes. A disagreement between multi-Virtual Organizations (VOs) might happen in the security aspect for the policy framework. Mobile grid services offer the ability to move jobs, data and application software from nodes to nodes during jobs' execution in the grid environment. It has also solved some problem in finding suitable resources for the jobs. To facilitate the ability to improve mobile resource sharing between multiple heterogeneous VOs, a policy management framework is needed to support the heterogeneity in the policy frameworks in different domains under different administrators.

Such a system should take into account user preferences. Few methods consider user preferences into their policy management frameworks. To date, not enough attention has been paid to policies that deal with such concerns. Most existing grid systems only support limited types of policies (e.g. CPU resources). A few frameworks consider enforcing data policies in their architecture [44], [60],[81]. We propose a policy-managed grid environment that addresses the user-submitted policy, data policy, multiple VOs, as well as the specifications and enforcement of user preferences and resource policies

in a grid environment spanning over multiple VOs.

In this research new policy management tools have been introduced that address mobile resource sharing and data movements within the grid. We present a dynamic and heterogeneous policy management framework that gives a clear policy definition about the ability to move jobs, data and application software from node to node during jobs execution.

1.2 Research Methodology

The research method applied in this research is a standard scientific research system, which comprises the following stages:

1. **Literature Review**

   The research literature stage expressed the research question by arranging the data and then examining and studying this information.

2. **Modelling**

   Modelling stage is to evaluate and analyse the problem articulated in the research questions. This model consists of the architectural and computational grid design.

3. **Algorithmic Development**

   In this stage, a new way with its algorithm, has been created to deal with the different concerns involved.

4. **Prototyping and Evaluation**

   In this stage, the prototype related to the model has been created. The experimentation progress has been performed and the results have been gathered and evaluated.

1.3 Research Questions

Traditional authorization policy management frameworks work well for a single VO where the contributing hosts grant the permission to follow a global authorization system. However most policy management tools do not provide support for sharing mobile resources between multiple heterogeneous VOs. Therefore; the research question is:

- **How does the grid interact with policies for different domains and organizations in the case of Mobile resource sharing and data movements?**

This question is divided into a number of subquestions. These are:

1. How to introduce a policy framework that supports a multi-organization environment over different domains?
2. How to introduce policy management tools that provides support for sharing mobile resources between multiple heterogeneous VOs?
3. How to design a policy framework that can support the user policy in its final decision?
4. How to enforce data policies within such a framework designs?

1.4 Research Contributions

The major contribution of this thesis is our new framework for Policy management that support the following features:

1. **Supports a multi-organization environment with different domains.**
   Grid infrastructure allows contribution and sharing resources at the level of a Virtual Resource (VR). The VR can be one device, a group of devices or a virtual partition on the correspondent device. Each grid institute has many VRs that are invited to participate with other contributors in the Virtual Organization (VO) [113]. Our framework uses well-established concepts from [121] that deal with multiple VOs (as explained in chapter two). Our contribution in this section is

proposing extension to the framework in [121]. The framework in [121] does not take the mobility technology in its design. Also the user preferences and data policy enforcement in the finial policy decisions have not been considered in its construct. Our framework added features for supporting grid user preferences, along with enforcing policies for data movements and resource mobility feature within different domains under different administrators. This extension is introduced in chapter three and has been published in the Communications in Computer and Information Science Conference (CoNeCo2011) [15] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

2. **Provide clear support for sharing mobile resources between multiple heterogeneous VOs.**

   Mobility is the ability to move physical or virtual computational resources (software code, data, running objects and mobile agents) from one site to another through a local or wide network. The advantages of such technology are increased resource utilization, enhanced organization between services and resources [17] and improved grid service flexibility [117]. Using the resource mobility feature, services can move across the grid to obtain data from grid nodes, implement their jobs on those nodes and pass the results back to their original nodes. The requirements of privacy and security apply to the mobility in secure grid environment are one of the important demands for both grid user and grid resources. Our contribution related to this section is introducing our new policy framework that provides resource mobility by using mobile policy agents that organizes the mobility for data, jobs, and application software within the grid. This framework is introduced in chapter three and has been published in the Risk and Security of Internet and Systems Conference (CRiSIS), IEEE Computer Society, 2011 [14] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

3. **Enforce user preferences in its final decision.**

   Before the users can submit their jobs or run their applications on a certain source or system they may need a guarantee that this source or system has not been compromised, which could subject their own application or data to being stolen. Or they may ask for certain users to be allowed to access the service or their data. These security issues should be taken into consideration when designing such a Grid system [65]. Our contribution in this section has been introduced in chapter three, where we propose an extension to a framework in [121] so as to be able to provide the features of supporting the grid user preferences in the final decision before applying the resource mobility feature. This framework has been published in Communications in Computer and Information Science Conference (CoNeCo2011), Risk and Security of Internet and Systems Conferenc (CRiSIS2011) and the International Journal of Computer Networks and Communications (IJCNC).

4. **Designing Enforcement Mechanisms for Data Policies.**

   Data is a part of information saved in a grid node, and is utilized by application software to accomplish specific jobs. The application software can then reach data whether on a local node or remotely. The data may have been kept on one or more nodes, or it may arrive with the user job. Not a lot of researches have given enough attentions to enforcing policies and handling data within the grid environments. Our contribution in this section is introduced in chapter three, where we propose an extension to the framework in [121] so as to be able to enforce policies for data movements before applying the resource mobility feature. This framework has been published in Communications in Computer and Information Science Conference (CoNeCo2011), the Risk and Security of Internet and Systems Conference (CRiSIS2011) and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

5. **Present a new language that supports and expresses the new policy framework.**

   Our grid system has been simulated using the Jade, which is a software framework implemented in Java language that allows agents to execute tasks defined according to the agent policy. In order to build the proposed grid system we designed interfaces that help the grid administrators to build their grid with all of it's resources. These interfaces give the user the ability to describe and send their jobs to the grid. These are been introduced in chapter five and six. The language that is used to design these interfaces is Java language. When grid administrators or grid users submit their requirements by using the previous interfaces to be simulated by Jade, our system converts these requirements to an External-JSDL that express the users' jobs requirements and can be understood by the grid environment no matter in what domain the resources lie. The External-JSDL language is introduced in chapter five. The proposed system converts the External-JSDL requirements to an Internal-JSDL; as a language that is used to communicate between the resource broker and the grid nodes, and as between the grid nodes themselves in different organizations and across domains. The system then stores these language expressions as an XML schema in order to be retrieved later and sends them to Jade. The reason behind using XML as a language for grid and job requirements expression is that XML has many attractive attributes such as the simplicity in reading, understanding and processing by users and computers. The Internal-JSDL language has been introduced in chapter six.

6. **Introducing the Jade simulation environment for modelling the policy framework and evaluating the performance of the grid in the case of mobile sharing and data movements.**

1.5 Thesis Outline

The following is a summary of the study's organization:

- **Chapter 2:** Introduces related work to all main research matters presented in this thesis. It illustrates distributed systems features and architecture, including a short discussion of grid computing types and challenges. Mobility and policies are then outlined including their relationship with grid resource brokers. Finally we present the most obvious challenges that stand in the front of developing the grid environment.

- **Chapter 3:** Introduces our new framework for policy management, and presents the differences between our framework for policy management and other approaches in this field. It presents our architecture together with each of its components and their special functions. Finally, it describes how a single institute policy agent can connect to the main grid policy agent in order to fitful jobs. This chapter shows our contributions in supporting a multi-organization environment with different domains, providing clear support for sharing mobile resources between multiple heterogeneous VOs, thus supporting user preferences in its final decision and enforcing data policies in its designs.

- **Chapter 4:** This chapter presents our computational model and procedures by giving a description of its components and the interactions between them. It describes how the involved application software and/or data migrate from one grid resource to another depending on the policies. Finally describing the method that allows users to manage and monitor their jobs during run time through events.

- **Chapter 5:** Gives an overview of a language which is used to build our grid system and an overview of the External-JSDL language which is used to describe users grid jobs. This chapter shows our contributions in presenting the new language that supports and expresses the new policy framework along with introducing our External-JSDL.

- **Chapter 6:** Presents an overview for the Internal-JSDL, which is used to describe the interactions between the grid resource broker and the grid resources and between the resources themselves. This chapter shows our contributions by

presenting a new language that supports and expresses the new policy framework along with introducing our Internal-JSDL.

- **Chapter 7:** To introduce our simulation for mobile grid environment and to show its components and the way to launch and deal with the simulation.
- **Chapter 8:** To show the validation of this simulation and to evaluate our outputs and discuss the results.
- **Chapter 9:** Concludes the thesis and outlines future work.

Chapter 2

Literature Review

2.1 Introduction

A grid is a system that should have the ability to organize resources which are not under the subject of centralized domain, utilize protocols and interfaces, and supply high quality of service [58]. Grimshaw and others in [61] define grid computing as "coordinated resource sharing and problem solving in dynamic, multi-institution virtual organizations." These resources are various from a single machine, a group of machines or a virtual partition on the same machine [113]. Thus, the major advantage of Grid computing is the capability to organize and share resources [25, 90]. As a result of such technology many challenges stand in front of developing and make use of its resources. There are a lot of contributions to solve some of these challenges, one of these contributions is the resource mobility which has solved some of the lack in finding the suitable resources for the job, but not a lot of attention was given to the policy (aspect of security and privacy) in this solution. This chapter provides a background and an essential overeview of the existing state of grid computing, including research efforts to the mobility solution and the policy feild related to this issue.
This chapter is structured as follows:

- Section two introduces Distributed Systems and its Architectures

- Section three reviews the definition, forms and objectives of grid computing.
- Section four presents the structural design of grid computing.
- Section five presents grid resource broker (Scheduler) including tasks and schemes.
- Section six reviews mobility in grid computing.
- Section seven reviews type of security solutions in grid computing.
- Section eight presents policy definition and category.
- Section nine describes grid technology infrastructure.
- Section ten introduces languages for describing grid jobs.
- Section eleven reviews the dynamic policy management framework that has been introduced by Globus toolkits.
- Section twelve present active network technology used to support grid system management.
- Section thirteen describes the recent used policies for data movements.
- The finial Section reviews the grid computing challenges.

2.2 Distributed System

H. Attiya and J. Welch [18] describe a distributed system as "a collection of individual computing devices that can communicate whith each other." They describe many advantages of utilizing the distributed systems such as increased reliability, information exchange, resource sharing, and increased performance.

Distributed systems are classified into two category according to number of nodes in the system. The first one is the static distributed systems which can be defined as systems with a fixed number of nodes. As the number of nodes in the system is constantly changing dynamic distributed systems appears as a second category of distributed systems. In the last part processes can join and leave the ongoing computation at any

time. Therefore, the set of processes in the system may change from one moment to another. Peer-to-peer computing is a clear paradigm of dynamic distributed systems. In spite of the different architectures between the existing distributed systems, almost all of them sharing the same unique characteristics which make them ubiquitous today through business, academia, government and home. Some of these characterstics include; separate address spaces, communication latency, coarse-grained concurrency, partial failure and variable configuration.

2.2.1 Distributed Systems Architectures

Researchers have become significantly interested in dynamic distributed systems because of the ability to solve large-scale problems requiring enormous computational power. Many methods of distributed computing such as client-server architecture, peer to peer computing, cloud computing and grid computing have been proposed. The architectural model is a way to describe the elements in a system and to describe how these elements co-operate and act with one another in such a system [36].

2.2.1.1 Grid Computing

Grid computing is defined in literature as "systems and applications that integrate and manage resources [1] and services distributed across multiple control domains" [82]. A grid is a system that has the ability to organize resources that are not under the subject of a centralized domain that utilizes protocols and interfaces, and supply high quality of service [50]. A. Grimshaw and others in [61] define grid computing as "coordinated resource sharing and problem solving in dynamic, multi-institution virtual organizations." The term 'Grid' obtains its name from the expression 'power grid', referred to a grid electrical power. The structure of an electrical power grid and grid computing generally make it possible for an user to use electricity (data) by simply plugging into a wall socket (network), without being worried about where and how the electricity (data) being used is generated [21, 79]. The grid should have the ability to enhance not only the systems performance and job throughput of the applications but also to

[1] Resources refere to computing and management resources such as computer, software applications, etc.

increase the utilization scale of resources used by employing effective resource management methods to the huge amount of its resources [7, 100]. Ian Foster [52] defines a grid by three main features:

- It organizes resources that are not dependent on centralized management. A grid combines and organizes resources and clients that exist within various control domains. The grid model allows the management and sharing of a huge number of geographically discrete heterogeneous resources.
- It employs "standard, open, general-purpose protocols and interfaces". A grid is constructed from various functional protocols and interfaces that can be employed in the grid's architecture.
- It shoud have the ability to provide various qualities of service, such as qualities security and response time.

Grid Aims and Features

Grid computing generally has the following objectives [51, 21]:

- **Sharing of heterogeneous and distributed computing resources which are owned by various domains.**

  Grid computing is the collection and accumulation of a set of shared resources such as storage systems, data sources, super-computers and management schemes that act like a network of computation [8, 84]. It supports and coordinates the sharing of heterogeneous resources that are distributed across multi-organizations.

- **Utilization of unexploited resources.**

  In most Organization there are large numbers of unexploited computing resources. Nearly all of these resources are active not less than 5% of the time. In addition, in some associations these resources are relatively inactive. Grid computing is built to take advantage of these unexploited resources and improve resource usage. Besides that, clients can rent the resources that exist in the grid to perform their computational jobs rather than buying their own expensive resources.

- **Simplify the cooperation between various organizations.**

  Another feature of grid computing is preparing the required environment to make it easier for different heterogeneous organizations to share and collobrate. This can be done by granting a direct access to computers, software and data storage.

- **Single login service.**

  One of the important features of the grid system is the ability to use a single login service for secure access to grid resources. This allows secure access to any information anywhere over any type of network. This is accomplished by presenting access control methods which controls these resources.

- **It is constructed to resolve vast problems.**

  Grids are designed to utilize unexploited resources, by employing a large number of these resources; grid is able to solve a huge problem such as weather forecasting, university experiments, etc .

- **More efficiently in getting and delivering results.**

  The other feature of grid computing is obtaining a result quickly and more efficiently. That's because of using parallel processing or having high capability devices.

Table (2.1) explains the differences between the three types of distributed networks [33].

2.3 Grid Categories

As grid computing becomes widespread, in many areas, grids have generally been classified from the viewpoint of application as well as their topology [73]. Grids are classified into solution and scale categories.

2.3.1 Solution Category

Grids are categorized into three groups according to the solutions they aim to present [76]: Computational grid, to support access to heterogeneous resources, Data grid, to

Table 2.1: Grid Computing vs. other Distributed Systems

Distributed Computing Environment	Control/management structure	Security policy structure	Typical users
Grid computing	1. Some centralization (because the present of resource broker/schedulers). 2. Some standardization be presented (such as the Global Grid Forum standards)	Complete security policies can and do present.	Users/resource owners are parts of many groups or organizations, or may be private owners.
Centralized network computing	1. Steady architecture 2. Several topologies exist (bus, token ring, star, etc.) 3. Administered and managed by a single entity.	High-level security policies	Members of a single group or organization; network administrator has control to all of them
Peer-to-Peer (P2P) computing	No centralized management structure	No centralized security policies; on the other hand single users/resource holder may have local security policies.	Part of many groups or organizations, or may be private owners.

supply data services such as data management, data access and storage, and finally Service grid, to provide services not supported by any single resource. An explanation of the functions of each follows.

2.3.1.1 Computational Grid

A computational grid is a group of computing resources that can be represented by computers over multiple domains and locations with different administrative domains and owners. The aim of this grid is to carry out large scale of applications using high performance servers. Jobs involved in this category of grid are those that present huge problems. The main features of these grids are the speed and reliability of networks, as well as the use of multi-distribution protocols which enable grid users to remotely

exploit resources owned by different suppliers.

2.3.1.2 Data Grid

In this kind of grid, a huge amount of data is spread, or in some cases duplicated, to remote sites. Generally, a data grid stands for a system providing services for storage, discovery, handling data and offering way to approach groups authorized to share it. That is, data grid offers the basic infrastructure for creating repositories for data that are spread over multi heterogeneous networks [76]. The aim of data grids as mentioned in [40] is to combine heterogeneous data archives into a distributed data management "Grid", with the purpose of identifying services for high throughput, distributed, data-intensive computing, and to allow users to obtain related data from the distributed databases. Data grids are consistent with computational grids and can combine storage and computation processes.

2.3.1.3 Service Grid

This type represents and allows a collection of services to be offered from a group of resources. Service grids can be divided into three categories: "On-demand" Service grids allows real time interaction, "Collaborative Service" grids combine multi-resources to supply new services, and "Multimedia Service" grids provide the infrastructure for real-time multi- media applications.
This research is considering and concerned with both computational grid and data grids.

2.3.2 Scale Category

Along with the former categorization, grid computing can also be categorized into four different applications according to scale (scope) and size [54]:

2.3.2.1 Cluster Grids

This type of the grid can be considered as the smallest grid in size and range. It aims to increase user job throughput by combining data and services so as to maximise the use

of computing resources. The grids which are designed to resolve problems for specific groups of people within the same department is considered as an example of a cluster grid. So, cluster grids can work within heterogeneous systems involving mixed server types, different operating systems and different work loads.

2.3.2.2 Intra-grids

Intra-grids consist of inter-connected clusters. Linking clusters allows the establishment of enterprise grids. Intra-grids allows sharing a set of resources subjected to common policies without essentially having to consider the security and global policy management issues for the whole grids.

2.3.2.3 Extra-grids

Interconnected cluster grids along with /or intra-grids, create another type of grids called extra-grids, which is geographically distributed between enterprise organisations. Therefore, these types of grids have several security domains; every domain has its own access policies. In this grid implementation, Virtual Private Networks (VPN) is used to make resources available to grids users.

2.3.2.4 Inter-grid

Sometimes it is called Global Grids. They are sets of Intra-grids and cluster grids connected by the internet. It can be used in the academia sector, where team groups may be a member of collaborating but geographically distributed systems.

2.4 Grid Architecture

The usual distributed methods do not provide an integrated method to access the large range of necessary services and resources in the grid, and they do not have the flexibility and management needed to allow the type of resource sharing required. In [54, 66, 62], the grid gives priority to interoperability, as it is critical to guarantee that virtual organization users can dynamically share varied and unutilized resources. The grid

infrastructure is built on a standard open architecture which simplifies interoperability, extensibility, portability and code sharing.

2.4.1 Grid Architecture Layers

This architecture manages elements into layers, as shown below in Fig (2.1) Elements within each layer share general attributes, based on the capabilities and activities of any lower layer.

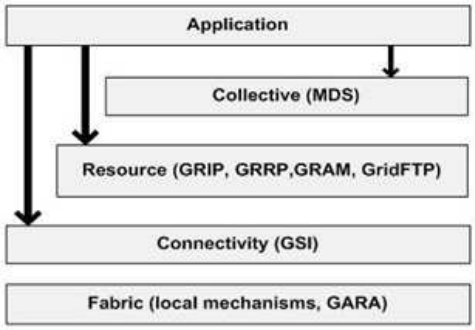

Figure 2.1: The Layers of Grid Architecture [54]

- **Fabric Layer**

  The fabric layer contains the resources in the grid. The resource varies from a single machine, a group of machines or a virtual partition on the same machine and can be either a logical resources like a computer cluster, distributed file system or distributed computer pool, or a physical resources such as network resources, sensors and storage systems. This layer is built on the actual resources and grants the access to it. Moreover, it applies the basic mechanisms that let those resources to be contacted and used.

- **Connectivity layer**

  The connectivity layer offers the basis communication and authentication protocols needed for network transactions over a specific grid. These protocols supply cryptographically protected mechanisms for validating the grid users and resources. A lot of communication protocols in the this layer are derived from

TCP/IP protocols model such as IP [69], TCP [68], UDP [94] , ICMP [95] and DNS [88].

- **Resource Layer**

  This layer is based on the connectivity layer, that executes protocols, that allows the utilization and sharing of a single resource like the Grid Resource Access and Management protocol (GRAM) which is used to assign and control resources. There are two basic protocols in this layer: Information Protocols which are used to query the situation of a resource by request fabric layer functions to control and access resources, and Management Protocols to negotiate gain access to a resource.

- **Collective Layer**

  This layer allows protocols to interact across sets of resources. More specifically, it concentrates on the managing of various resources. It consists of directory, scheduling, co-allocation, monitoring and diagnostics, brokerage, data replication, community accounting, software discovery and payment services.

- **Application Layer**

  This is the top layer in the architecture, and includes the user applications that run in a grid environment. It contains the languages and frameworks. These frameworks can describe protocols like services, Simple Workflow Access Protocol (SWAP) [107], or in some cases an Application Program Interface (API).

2.4.2 How a Grid Works

Grids depend on middleware, which is enhanced software and/ or hardware that guarantees smooth communication between distributed resources. Grids employ effective discovery services that determine unutilized resources in the grid so as to utilize them. Users should have the sufficient authorization to use the grid through software interfaces working on their personal machines. After authentication, the user will be able to express the job to the grid resource broker (scheduler), which is the core of the grid.

The resource broker will locate free resources that can best match the user's requirements and job (or users' applications) conditions by contacting both the information service (IS), to retrieve information about software and hardware presently available, and the replica catalogue (RC), to find out the location of needed data. Once the application has chosen the suitable resources for the job, or has made advance reservations on the chosen resources, the job is sent to those resources for implementation. After the resource broker submits the results back to the user as shown in Fig. (2.2) below. All of these processes are ca trried out ransparently of the user, who sees the grid as a single large and powerful computer [76, 91].

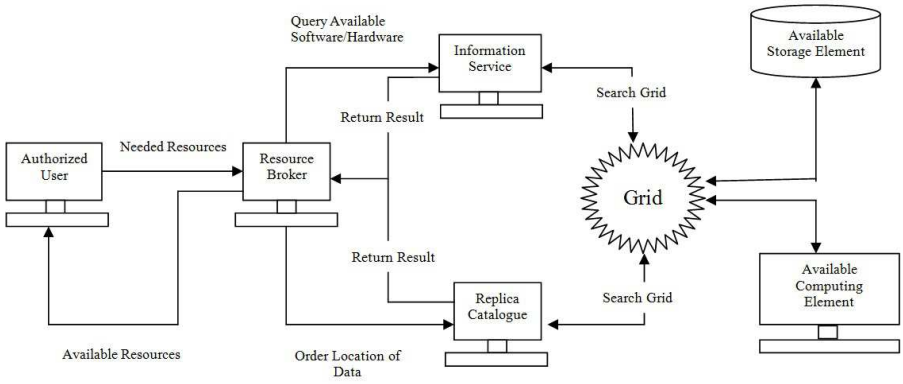

Figure 2.2: How Grid Works

2.4.3 Information Service

Information service is a crucial element in grid computing. It is a directory service that holds information about the resources in the grid and the entire grid activated jobs running on those resources. This information can be static or dynamic information related to the resources available time, disk space, the job presently running, application software, and policies.

2.4.4 Replica Catalogue

This is also an important component for the grid, because it presents information and helps in accessing the stored data in the grid. It determines the places of data in the

grid, updates data resources and maps logical file names to the actual physical places on grid resources. A resource broker communicates with a replica catalogue to ask for information about data location and the access control needed to use this data.

2.5 Resource Broker (Scheduler)

Some people called it a "Scheduler". The resource broker is one of the major grid elements. It performs significant functions in building a valuable grid environment by scheduling user jobs onto grid resources to reach a specific targets such as cutting the execution times and communication delays, raising the resource exploitation and reliability, distributing jobs across resources without depending on a particular resource, and loading balancing. The main jobs for the broker are to discover and choose suitable resources for jobs by sending jobs input files to the resources, monitor jobs and send job outputs back to users. The following section explains these tasks.

2.5.1 Resource Broker Tasks

- **Resource Discovery**

  The task of discovery is that of choosing a set of authenticated free resources in the grid. This set is usually acquired by exploring its database including data about the resources. Resource discovery methods employ a single database (the centralized approach), or a group of databases (the distributed approach); the main job for these database is to retrieve information about logical units such as application software, policies, operating systems and data and physical units such as CPU speed and architecture, current loads, and networks, to choose the resources that can fit the application conditions. Monitoring and Discovery Service (MDS) in Globus is an example of this database [47, 67, 27, 32, 85].

- **Resource Selection**

  When the resource set has been collected, the best resources that fit the user's conditions, like cost, are then chosen.

- **Application Execution**

  After the job and resources have been chosen, job input files are sent to and executed on the resources. Unexpected situations that need addressing may happen at some point in job runtime, therefore the job monitoring job is to verify the job execution and spot failures or unpredictable clashes. As soon as the job is done, the broker notifies the user. Network Weather Service (NWS) is an example of such monitoring [64].

2.5.2 Resource Broker Schemes

In grid environments, three types of resource broker schemes can be applied: centralized, hierarchical and distributed scheme. The following illustrates each one of these types.

2.5.2.1 Centralized Scheme

In this category the resource broker is a main machine (a server) holds data about all the resources in each domain. All requests are sent to the resource broker, which sends it to the appropriate resources according to the information available. Figure (2.3) below illustrates the architecture of centralized schemes. With a centralized resource broker the decision making becomes faster and better, since they have all the needed and the latest information about the resources. As a result the execution in this schema should be faster than other schemas. Alternatively, this kind of schema does not consider scalability because of the growing size of the environment that they handle, which means all the requests are being submitted to the same broker. Thus, applications in some situations are affected from long access delays. For the same reason, if the resource broker fails, the communications between the users and resource providers break off [83, 55, 10, 42, 89, 96].

2.5.2.2 Hierarchical Scheme

In hierarchical schemes the central resource broker communicates with local resource brokers, as shown in Figure (2.4) below. All jobs are sent to a central resource broker

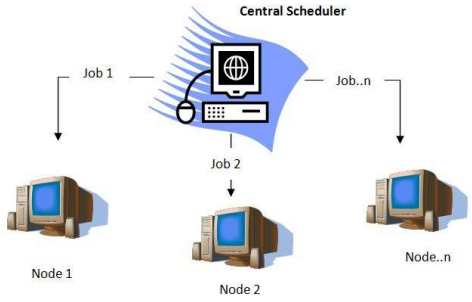

Figure 2.3: Centralized Scheme

who submits them to the domains that can fit their conditions. At this moment, the central broker has no direct power over those jobs. A feature of such schemes is that each local domain can use its own resource broker policy. On the other hand, the job cannot be moved or reallocated to another resource at different domains, even when a better resource is found [34, 123].

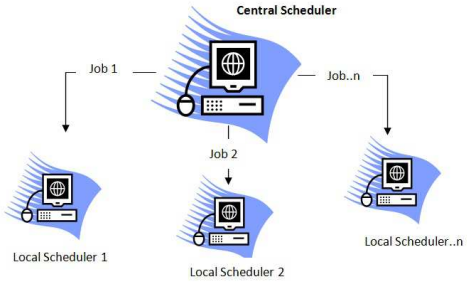

Figure 2.4: Hierarchical Scheme

2.5.2.3 Distributed Schemes

In this schema there is no central resource. As an alternative, each local domain has its own resource broker, as shown in Figure (2.5). Each domain inquires of other domains periodically or when an event occurs, to get information about the status of resources exist in other domains. When a job is to be executed, it is sent to the local resource broker which exist in the same domain and then to an appropriate local resource or to another more fitting resource in other domains. The features of this scheme are: reliability, scalability, smoothness of implementation and the problem of a single point

of failure, but in some situations the allocation can be instable and take a long time [89, 43, 109, 106, 26].

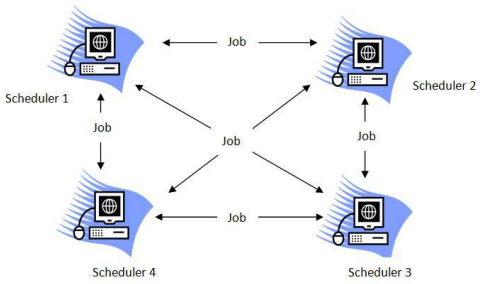

Figure 2.5: Distributed Schemes

2.6 Policy

Policies are groups of regulations, standards and practices written by one or more owners of jobs or administrators of resources, about how their resources or jobs can be handled and used. It decides how a job should be done, how security is applied in a domain and how an organization organizes, secures and distributes their resources. Policy can be static or dynamic; in the static policy data is 'read only', while in the dynamic policy data can be migrated, written, read and copied. Policies can be divided into user, grid and resource policies.

2.6.1 User Policy

Before the users can submit their jobs or run their applications on a certain source or system they may need a guarantee that this source or system has not been compromised, which could subject their own application or data to being stolen. Or they may ask for certain users to be allowed to access the service. These security issues should be taken into consideration when designing such a Grid system [65]. These security issues should be taken into consideration when designing such a Grid system [65].

2.6.2 Grid Policy

This is a grid management approach that describes how to select the grid resources. It is used by resource brokers who impose it. One example would be the order to select the lowest load resource from the appropriate resources set in order to perform the job.

2.6.3 Resource Policy

This describes how grid resources are utilized. Resource owners have the privilege to decide the policies for control of their resources. Resource policies are saved in the information service (IS) and are imposed by resource brokers; they can be static or dynamic. In static policy the node can only obtain jobs with their associated data (i.e. user data) in order to perform it in that node; it does not let this data or application software to migrate to other nodes in the grid system. To be more specific, the node data is just read only. While the dynamic policy grants resource owners more choices to choose their own polices for all node elements by deciding the policy for each element separately. The node has many elements including data and application software.

2.7 Mobility

Mobility, is the ability to move physical or virtual computational resources (software code, data, portable notebook PC's, running objects and mobile agents) from one site to another through a local or wide network. Mobility is a wide idea used in distributed computing. With the assist of the mobility, the services can move all over the grid to obtain data from grid nodes, implement on those nodes and carry the results back to their original nodes. This can enhance the utilization and grid services flexibility [117]. Mobility may be divided into personal, computer and computational mobility. In personal mobility; the grid users can do the job at sites remote from actual physical hardware, without having to move jobs around with them. They can launch a job in one site and move it to another place in the world no matter the machine type, such as web-based email accounts. The second type of mobility is Computer mobility, which is the transfer of an actual part of computer hardware such as PCs notebook and Personal

Digital Assistants (PDAs) from one site to another.

The last one, in which this thesis is interested, is the Computational mobility, which deals with the movement of software [108, 48]. Computational mobility can also be called a control migration, data migration, link and object migration [28]. Control migration offers moving execution control such as Remote Method Invocation (RMI) and Remote Procedure Call (RPC) from one system to another and back again. Data migration lets the data requested by the job to be sent on through the network. As an example of such mobility is Java RMI call methods. Link migration, is referred to the ability to move objects (codes) between multiple servers. Code migration (mobile computation) affords the ability to connect software elements at runtime. In other words, software elements can move around and execute on various servers across the network. From the aspect of execution state, code mobility (migration) can be divided into two categories, weak and strong mobility [30].

2.7.1 Weak Mobility

Weak mobility allows code to move through networks. In some cases the codes have initial data assigned but without execution states (for example the first state of the computation at the first node has not been moved). An example of a weak mobility system is Code-on Demand (CoD) and Remote Execution Evaluation (REE) [48, 104]. Remote evaluation developed originally from the client-server and virtual machine styles [56]. A user element knows how to control and manage the remote job; the user submits the command to a server element at the remote site, which in turn implements the code using the resources offered there. The user then obtains the result from the server. The remote evaluation method presumes that the code offered will be implemented in a secure environment so that it will not disturb other users operate on the same server outside the capability of the resources being used [46]. One of the main features of remote evaluation is the capability to modify the services as server elements, which enhance extensibility and customizability and improve efficiency when the code can change its actions to the environment within the server. Grid computing provides a good example of REE. Code-on-Demand style schemes are used when a user element

has access to a group of resources but does not know how to process them. For that reason it sends a request to remote server to obtain the code of ”know-how”. When the code is arrived, the user executes it locally. A famous example of Code-on-Demand is Java applets.

2.7.2 Strong Mobility

Strong mobility is the ability of a computational environment to mobile the code and execution state (the context of execution) to start again at a new resource. The execution state comprises of running code, saved processor registers, program counter, local variables and return addresses. A set of organizing execution controls operates on the user's node and then accesses the remote resource by the invocation of a remote process. One of the most important features of strong mobility is that jobs can choose to migrate between sites while it is processing information. The reliability challenge of partial failure is decreased because the job state is only in one site at a time.
During the mobility, if the policy of the target node allows resuming the job from the point of frailer or from the last point of operating before the migration then the policy in this case does support the strong mobility and it is called a strong policy. Otherwise this policy is considered a weak policy as it supports only the weak mobility (staring the job from beginning).

2.7.3 Grid Mobility

In some situations the resource broker has to reject some jobs because one or more of the following situation:

- The resource required to match the job conditions is occupied at this time.
- The resource that matches the job hardware conditions does not have the required application software.
- The resource that matches the job hardware and application software conditions does not have the needed data.

As a result, the Mobility has created a new environment that can solve these rejected job cases. The grid mobility allows jobs, application software and/or data to migrate from one node to another in the grid environment so as to adapt the resources needed to fit the job requirements. The following explain each one of these motilities.

2.7.3.1 Job Mobility

The grid mobility technology allows the job and its execution state (i.e. the context of execution) to migrate from one resource to another and restart on the new one in order to fit the job conditions and requirements.

2.7.3.2 Data/Application Software Mobility

In this stage the data and/or application software are allowed to to migrate from one node to another in the grid environment so as to adapt the resources needed to fit the job requirements. These data or application software can be copied or moved depending on the policies of the resources that own them.

2.8 Security

Any secure grid environment should provide mechanisms to secure authentication, resource protection, authorization, secure communication, data transfer and encryption [77]. Grid security has several security challenges, involving coordinating user identities among local and wide networks, dealing with the variety of local security techniques for either resource or user, trust relationships between resources and users, end-user key, credential organization, and supporting security to resources in opposition to unsafe actions from grid users [53]. In [33], Erin Coday and others mention two different types of solutions that may be used for grid security: System solutions and Behavioral solutions.

2.8.1 System Soultions

This part of the classification handles solutions which focus on employing the hardware and software of a grid system directly to solve security problems. It can be divided into: system security for grid resources and intrusion detection systems (IDS). Security for grid resources depends on using technology, rather than policies. This can be done by separating the portion of the resource that is contributed in the grid from the portion of the resource that the owner wants to keep private. As an example of this solution is the sandbox, which stores applications files and data files encrypted on disk, so nobody can reach or use the application who does not have authorization to access it. (IDS) depends on monitoring resources and detecting any intrusion that may take place on one node and inform other nodes about possible misuse of resources by an intruder. The problem with this solution is the detection and reporting of an incident is dependent upon the relationship between nodes. Some nodes can share information with each other because of their relationship, but might not do it due to the cost of sharing.

2.8.2 Behavioural Solutions

These solutions depend on security by policy and human action in place of security using some hardware or software products. Such an examples of these solutions are: Accountability, group management and trust. The following are three types of Behavioral solutions; the first one is the Comprehensive Policy Controls which depends on the policy definitions, which this thesis is interested in, and the other one is the Trust-based security solutions which applies the trustworthy theory and the last one is the Hybrid solutions.

Comprehensive Policy Controls

This part achieves the security through policy definition. These policies mange all aspects of grid computing, involving sign-on procedures and access control, authorized user selection, and local/wide security settings. Comprehensive policy uses policy-driven access control to support groups that are geographically spread [39]. That can be achieved by the following steps:

1. For each group there is an administrator that control access to this group by issuing certificates proposed for this issue.

2. Policies are deployed to various groups in the domain by policy objects. Applying this policy is left up to the individual entities, allowing policies to be enforced over heterogeneous systems.

3. All groups swap and update policy information with one another, producing a wide global policy system that covers the whole grid.

4. This integration and collaboration of policies used by one node or group of nodes with policies inherent to the whole grid achieves the essential global grid security, while keeping the autonomy of grid systems.

There are many advantages and disadvantages related to this aspect of grid security. It allows for the use of rules and policies to achieve the aim of a secure grid network [53, 102]. This support performance and heterogeneous nature of the grid. But authentication from global to local nodes cause the need for a mapping table from global grid to local IDs which could also become prohibitively large as the grid grows in range.

Trust-Based Security Solutions

A user or resource of the system can make better judgments about dealing with other entities if they know the reputation of those entities in the system. The trust level between entities in the grid can be made up from many different factors including reputation, direct experience and time since the last participation with the entity in the grid [20]. Direct trust (trust-based on relationship between entities) is asymmetric, so each entity (resource or client) make a decisions how trustworthy the other should be. A trust agent estimates the level of trust based on the straight trust relationship, and on the recommendations from other nodes in the grid. Each resource has its required Trust level, or the least amount of trust level a user should have to utilize a resource. Grid users also set required trust levels for resources they want to use. Each local resource has a trust penalty levels suitable to the causes the offense had on its system [33]. Trust levels can be easily updated, and nodes/users can inherit these levels when joining a

system. Furthermore the domain can increase its required trust level to maximum in order to enforce improved security 100% of the time [19].

Hybrid Solutions

In some cases, it is better to use both system-based solutions and behaviour-based solutions to solve security issues in the grid. Therefore, it is more suitable to create a Hybrid Solution sub-category to address this issue, since it falls equally under System and Behavioural grid security solutions [33].

2.9 Policies

Policies are groups of regulations, standards and practices written by one or more owners of jobs or administrators of resources about how their resources or jobs can be handled and used. They decide how a job should be done, how security is applied in a domain and how an organization organizes, secures and distributes their resources.
Depending on the Globus Toolkit [13], before the job submission there should be many steps to authenticate the users who are asking to use resources [45, 118]. But after the authentication there are no further resource access restrictions on how to use the resources. This is known as "all or nothing".

2.9.1 Policy Categories

2.9.1.1 User Policy

Before the users can submit their jobs or run their applications (a group of jobs) on a certain source or system they may need a guarantee that this source or system has not been compromised, which could subject their own application or data to being stolen. Or they may ask for certain users to be allowed to access the service. These security issues should be taken into consideration when designing such a Grid system [65].

2.9.1.2 Grid Policy

This is a grid management approach that describes how to select the grid resources. It is used by resource brokers who impose it. One example would be the order to select the lowest load resource from the appropriate resources set in order to perform the job(s).

2.9.1.3 Resource Policy

This describes how grid resources are being utilized. Resource owners have the privilege to decide the policies for control of their resources. Resource policies are saved in the information service (IS) and are imposed by resource brokers; they can be static or dynamic. In static policies the node can only obtain jobs with their associated data (i.e. user data) in order to perform it in that node; it does not let this data or application software to mobile to other nodes in the grid system. To be more specific, the node data is 'read only'. While the dynamic policy grants resource owners more choices to choose their own polices for all node elements by deciding the policy for each element separately. The node has many elements including data and application software.

2.10 Grid Technology Infrastructure

The standardized architecture of the grid makes procedures and exchange parts easier between different organizations. Because grid tools and equipments are from multi-vendors, interoperability becomes important and high standards must be identified. For standardizing grid requirements, protocols and interfaces, the Globus alliance and Open Grid Forum (OGF) were launched, as described below.

2.10.1 Open Grid Forum

Open Grid Forum (OGF) [49] is a public society forum for discussing grid technology matters. The aims of OGF involve designing of open procedures for the improvement of grid agreements and specifications and create grid architecture documents and the most suitable guidelines. Many research groups within OGF have established various standards such as Open Grid Service Architecture (OGSA) to offer a service-oriented

view of the shared physical resources or services provided for theses resources, Open Grid Services Infrastructure (OGSI) to describe methods for establishing and organizing grid services, GridFTP and JSDL [6, 9]; a lot of other subjects are at present being worked on.

2.10.1.1 Job Submission Description Language

The Job Submission Description Language (JSDL) [6] is an XML-based description language proposed by (OGF) as a standard language. This aims to express conditions of computational jobs to be ready to transmit later to resources, more specific in grid environments. The design of this language makes it easy to be mixed with other languages to accomplish a bigger functionality. To smooth the progress of the illustration job conditions as a group of XML parts, JSDL includes a vocabulary and normative XML Schema. The motivation behind JSDL was the various job management systems used by diverse organisations; each system has its own private language for expressing job requirements and conditions. As a result, it became hard for a system to manage jobs and at the same time deal with other systems in different organisations. One example of such projects that uses JSDL in its systems is the Globus toolkit [13].

2.10.1.2 GridFTP

GridFTP [114, 12] is an extension of the standard FTP protocol used for grid computing. It is designed to supply effective and protected access and transport large amounts of data between multi-distributed resources in the grid. The FTP protocol was selected because it is one of the most widespread data transfer protocols and because it contains a lot of characteristics like; it is widely implemented, has a clear architecture, transparency, and its support for third party transfers. GridFTP offers a lot of advantages such as parallel and partial data and file transfer and enhance Grid Security Infrastructure (GSI).

2.10.2 Globus Alliance

Globus Alliance [13] is an international association of establishments conducting research for the enhancement of elemental grid technologies. Globus Alliance presents open source software named Globus Toolkit for creating grid environments and applications. The Globus toolkit supports a group of essential services required for grid computing. For examples: security, data management, execution managers and information services. These are illustrated in Table (2.2) and shown in Figure (2.6).

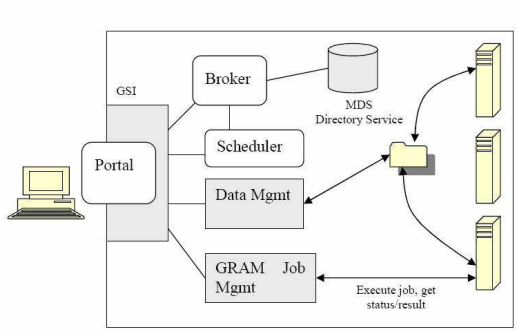

Figure 2.6: Globus Toolkit Architecture [13]

Table 2.2: Globus Services

Name	Service	Description
GSI	Security	Remote Authentication Services
GRAM	Resource Management	high Grid Resource Management
Data management	Transfer data	Manage data using GridFTP
MDS	Information	Grid Meta Directory Service
GEM	Executable Management	Managing location if executables

2.10.2.1 Security

Security is an essential element for grid computing. For any grid environment there should be methods to offer security, involving authentication, authorisation and data encryption. The Globus Toolkit provides strong security mechanisms by using Grid Security Infrastructure (GSI) component. GSI supplies a public key infrastructure for authenticated and private communication, in addition to authenticate users and guarantee that they are allowed to utilize the resources in a specific domain.

2.10.2.2 Data Management

Relocating data to a number of nodes within the grid requires secure and trustworthy mechanisms. The Globus toolkit has a data management element that offers such services. This component employs GridFTP [114, 12].

2.10.2.3 Execution Management

This is the centre of Globus Toolkit facilities. The Grid Resource Allocation Manager (GRAM) is an essential library service that supports remote implementation along with its status management. It can change a user demand for resources into commands that local computers can handle. In other words, it supports secure and trustworthy services to start jobs on specific resources, checks their status and recovers the results when they are done [73].

2.10.2.4 Information Services

Grid resources need services like Monitoring and Discovery Service (MDS) for locating and publishing their status and configuration information. MDS [73, 72] facilitate grid information services by supporting access to static and dynamic information related to resources. These locates the resources that fit the job conditions, as well as getting the state of those resources by using MDS components. These components can be summarized as following:

- **Information Provider (IP)** which translates the characteristics and status of local resources to the configuration described in the schema and configuration files. It consider an interface for any data gathering service that collects data about a specific part of a resource like RAM memory, disk capacity or CPU load.
- **Grid Resource Information Service (GRIS)** is a distributed information service that can respond to questions about a specific resource by pointing the queries to the underlying IP. GRIS stores and shows the information as entries.
- **Grid Index Information Service (GIIS)** is a warehouse that includes indexes of resource information registered by GRIS and other GIISs.

- **MDS client,** which is based on the Lightweight Directory Access Protocol (LDAP), is a client instruction to find resource information in grid environments.

2.10.3 The Grid and Industry (GridPP)

GridPP is a group of Particle Physicists and Computing Scientists compromise of 19 UK universities, Rutherford Appleton Laboratory and CERN (Conseil Europeen pour la Recherche Nucleaire) [4]. They have constructed a distributed computing grid crosswise the UK for particle physicists. The vision of this project is to establish and organize a computing resources for UK particle physicists using applications, open source software and middleware.

The idea behind created such a system is to share data, computing power and applications software, make use of resources at many institutes, conect main and small computer centres, ensure all data are reachable at anytime and anywhere and to be able to handle with varous management policies of varous centres [3].

2.11 Languages for Describing Grid Jobs

2.11.1 Job Description Language (JDL)

JDL [92] was presented by the European Data Grid [1] and is described by the Condor project [5]. It is a high-level, user-oriented language depending on the Classified Advertisement (ClassAd) language [97], an expression language permitting the user to show the resource conditions and job specifications. These specifications let fast and easy fit between requests and resources to implement the job on the right resource(s). JDL allows the user state the job elements (attributes) to help determine the suitable resource(s) to perform the request. These are job attributes such as job type, data attributes (like input data) and resource attributes (such as the number of requested CPUs).

2.11.2 Globus Resource Specification Language (RSL)

RSL is an XML based language used to express grid jobs which are going to be implemented on grid nodes. RSL specification has its private structure and includes a group of attributes and a group of operations that are merged to state the grid job. In RSL, the attributes can be used to express both job factors, like directory and name, and resource conditions, like number of nodes and machine categorize.

2.12 Dynamic Policy Management Framework

The Dynamic Policy Management Framework (DPMF) consists of three agents; Policy Agent (PA), Policy Management Agent (PMA) and Grid Information Agent (GIA). Figure 2.7 shows the hierarchical architecture of these agents.

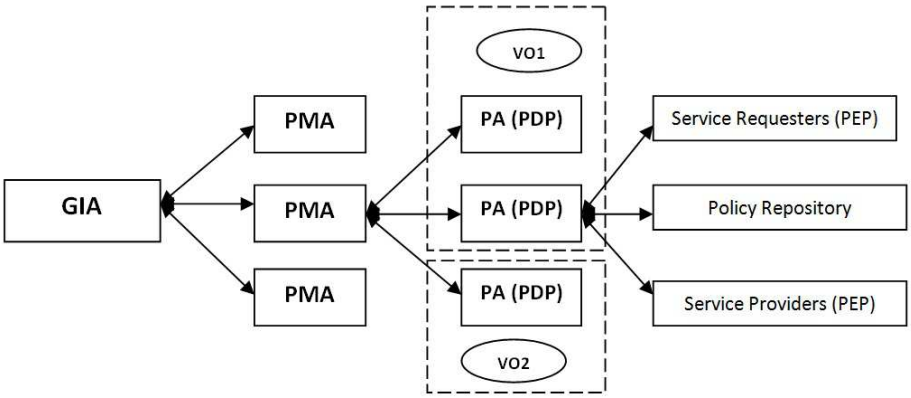

Figure 2.7: Dynamic Policy Management Framework

Each Virtual Organization (VO) should have at least one policy agent that has the ability to access the policy repository for that organization. Policy control between VO irrespective of the concrete organisation. For PAs of the same virtual cluster (the same policy frameworks) there should be a PA leader that coordinates other PAs in the cluster and at the same time performs a homogenous and a heterogeneous policy management across different policy frameworks, this agent is called (PMA). The (GIA) owned by the Grid administrator is responsible for providing PMAs with the necessary informa-

tion that is needed to perform the heterogeneous policy management across different policy frameworks.

When the PMA receives a request from one of the PAs asking for a service that is situated in multiple virtual clusters, a heterogeneous policy management mechanism takes place. This mechanism depends on converting the target service's policies into the policy model of the PMA. The PMA asks the GIA for the account maps and the policy schema maps for the target services. At the same time, the PMA asks the service requester for its authorization policies through the service's PA. When the PMA receives this information, it starts the policies conversion to the PMA policy Model. First it applies the Account Mapping; which is to make it possible for users, whether trusted or not, to access services on a remote VO. In this case a map mechanism should be applied to map those users to local accounts, and later perform the Policy Mapping to generate a inter-schema map, which maps the schema of policy model of the service's VO to that of the requester's VO. Or in other words; to convert policies between heterogeneous policy models to one that can be understood by the PMA. The PMA applies the conflict analysis mechanism on the policies of all target services to find suitable permissions for the service requester[121].

It can be seen that policy management is catergorized into inter-cluster heterogeneous policy management, and intra-cluster homogeneous policy management. To solve the conflicts between the same policy frameworks the intra-cluster hemogeneous policy management takes place, while the intre-cluster heterogenous policy management takes place in the case of the conflicts, between different policy frameworks.

2.12.1 Account Mapping

To make it possible for users, whether trusted or not, to access services on a remote VO a map mechanism should be applied to map those users to local accounts. If a VO decides to grant external users a permission to access it resources, it is necessary to initiate or assign local accounts for them so that they can only utilize permitted services [121].

Each VO initiates some accounts that can be used by external users or allocate some

active accounts to them. There are two types of these accounts; the first one for users from trusted VOs or trusted users. The second one is for users from un-trusted VOs or un-trusted users. As a result, if a remote user asks for a specific service from a VO, in this case the external user account is mapped to a local account which is initially created for this propose. The types of accounts are determined according to the trust relationships between the two VOs [122].

It is necessary for each PA who wants to join the Grid environment to provide the GIA with the mapping account information about its VO [121, 122]. The GIA then stores this information in its repository waiting to be retrieved from PMA(s). As shown in Figure (2.8) the account map information contains VO identity and the local account. When a PMA executes a policy decisions using the Heterogeneous Policy Management mechanism, it retrieves the account maps information from the GIA. By using the account maps, the PMA can choose the policies that are suitable for both the requester and the service provider for both different VOs in relation to the trust relationship between them. Figure (2.8) shows the generation of Account Maps.

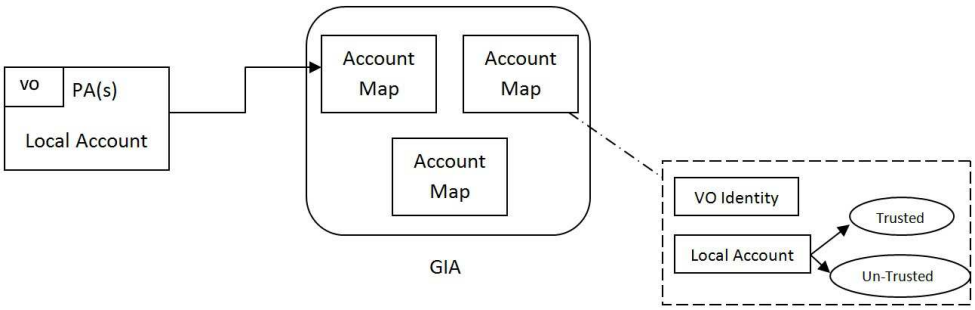

Figure 2.8: Account Mapping

2.12.2 Policy Mapping

Policy Mapping mechanism is used to convert policies between heterogeneous policy models to one that can be understood by the PMA. The Local Policy Schema (LPS) stands for the policy model for each VO. LPS describes the structure and components of the policy model [122].

The GIA gathers policy schema elements from VOs in the grid and sorts them into groups. Each group has meta-schema elements which contain policy scheme elements of equivalent or much related meaning. GIA collect these meta-schema elements and stores them as a Meta-Schema Taxonomy. Figure (2.9) illustrates the generation of Policy Schema Map which comes as a result from sending the PA its local Policy schema during the registration or after a change of policy model [121].

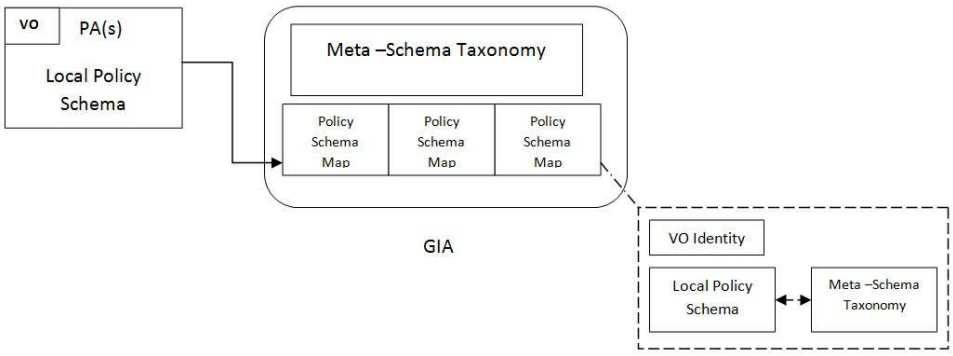

Figure 2.9: Policy Schema Map

When a PMA executes a policy decisions using the Heterogeneous Policy Management mechanism the policy mapping is taken place after the Account Mapping. PMA retrieves the Policy Schema Map from the GIA and then it produces Inter-Schema Maps for the purpose of conversion policies in heterogeneous policy models. From the Figure (2.9) each Local Policy Schema is mapped to Meta-Schema Taxonomy. The results show that PMA can produce Inter-Schema Maps by comparing the Meta-Schema Taxonomy for the policies that need the manipulation process. If there are two elements of the different local policy schema that are mapped to the same meta-schema element, then these two elements form a single mapping entry. Figure (2.10) shows the Inter-Schema Maps generation [121, 122].

Full mapping takes place if the all of elements of the source policy model is a subset of that of the target policy model. In the case of partial mapping, the un-matched elements will be hold. The PMA takes into account these elements as unrecognized elements. Once the permission condition for the authorization request is produced, the

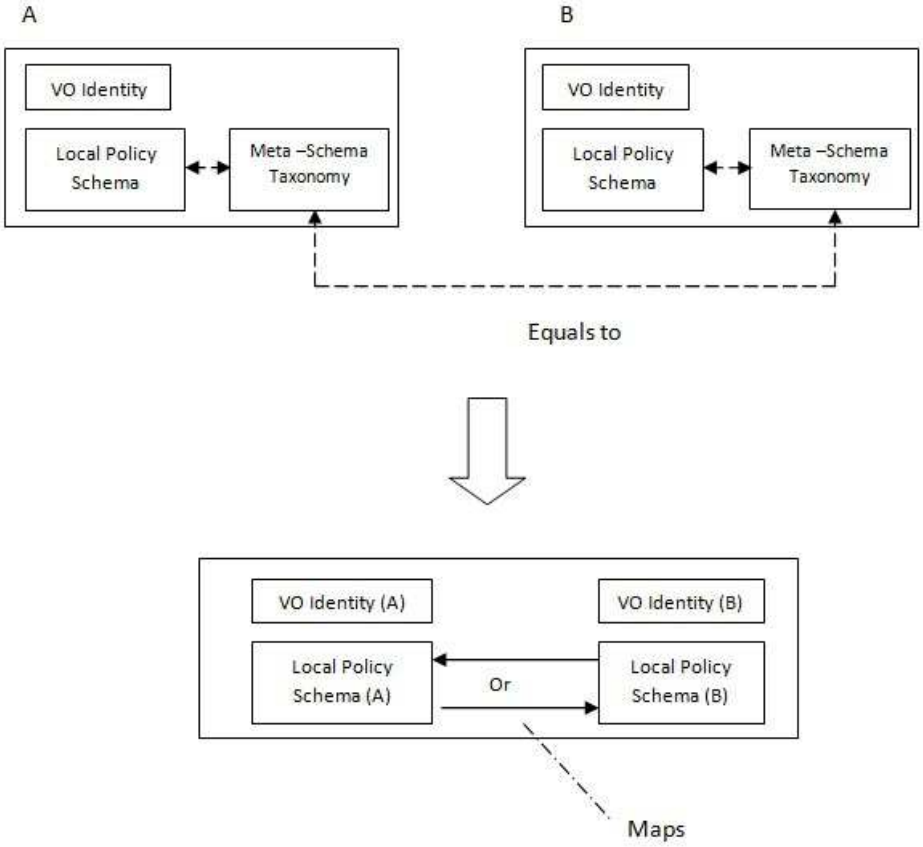

Figure 2.10: Inter-Schema Maps generation

unrecognized elements will be intersected to the permission condition. This is because the elements can change the permission condition [121].

2.12.3 Partial Policy Information

Each Virtual Organization (VO) should have at least one policy agent that has the ability to access the policy repository. For PAs of the same virtual cluster, there need to be a PA leader that coordinates other PAs in the cluster and at the same time performs a homogenous and a heterogeneous policy management across different policy frameworks. In an ideal grid system each Policy Management Agent (PMA) is being trusted from all the PAs in the virtual cluster; that makes the conflict analysis between the user policy and the service provider's policy very clear to the PMA. In reality, not all PAs

do trust the PMA; in this case PMA has to do a conflict analysis with partial policy information. In other words PMA executes conflict analysis with the identified security policies, produces replacements for unidentified policies.

Chiu-Man and Kam-Wing Ng in [120] have mentioned ways of generating substitutions for unknown policies. This way depends on discovering a group of conditions which may generate conflicts. After that, these probable conditions are transmitted to the un-trustful VO (PA) for its examination. The conditions discovering is taking place in the policy template database which was created originally when the PMA was firstly created. This database includes a group of "common policy templates". These templates are produced by scanning policies of VOs (which trust the PMA) in the same virtual clusters. PMA should have the ability to swap policy template group information between trusting PAs.

The Policy Template model consists of Evaluation Set, Priority set and the normal policy components. Each policy consists of four components; Condition Set, Action Set, Extension Element(s) and Target Identity. The Condition Set is a group of conditions with (AND) or (OR) relationship(s). Each condition is a Boolean syntax which means the result are either true or false. The Action Set is a group of actions that take place as a result for the condition with an (AND) relationship; these actions can be performed concurrently or in one-by-one order. The Extension Element(s) is used to register elements which cannot be represented by previous elements. These extra rooms support the compatibility with various policy types, but to activate this component, all elements that are responsible for a conflict analysis task (for example: PMA, GIA, PA) have to be able to identify the extension element(s). The Target Identity is a credential to authorize a user, or just a login name in the local domain. The Evaluation Element Set keeps evaluation elements which are characteristics of the policy holder. These characteristics are described by the PMA. They possibly contain the type of service providers, the type of VOs, security levels, etc. The Priority Set keeps the order of priority of the evaluation elements.

The conflict between policies occurs when two or more policies have the same set of conditions but have opposite set of Actions. To be more specific have an opposite set

of authorization action. This is called a modality conflict [41]. The conflict detection can be classified into two types; the first one is related to the detection task including service providers on VOs which all trust the PMA. In this case the related PAs support the PMA with the necessary policies information connected to the requester identity. These policies information include a group of service providers and the VO domain. PMA looks at this information trying to select ones whose resource field is the target resource. Then PMA tries to detect any conflict in these policies by checking both the authorization signs and the condition sets. If the PMA discovers any intersections between these policies then a conflict is detected.

The second type of detection is the conflict detection with Partial Information. This type of detection takes place when some PA(s) do not trust the PMA. In this case, PMA gets the policies information related to the requester identity form the trustful PA(s). For un-trustful PA(s), PMA employs policy templates to produce substitution policies. The production goes into three levels; the first one is choosing the policies from the policy template database. This can be done depending on the Evaluation and Priority sets. First; PMA searches the database looking for the policies with the same priority sets for the evaluation sets and then does the second search. The second search has two search parameters; the evaluation sets plus the results of the first search.

The second level is executing a conflict analysis for all related polices (the selected policies from level one and the policies from the trustful PAs). The results of this level are policies that could cause conflicts to the service request. These policies may be stored in the Conflicts Policy Set (CPS).

The third level is sending the CPS to the un-trustful PA(s), this is done by the PMA, to see if the PA's VO domain and its respective service provider(s) have any one of these policies. If the answer is 'no' that's mean no conflict is found. If yes the conflict is found. If the PA doesn't reply to the PMA that's mean the information that has been sent to the PA is not enough for the conflict analysis.

2.13 Active Network for Grid Management

Active network changes the store-and-forward network into store-compute-and-forward [58]. The improvement in such a network allows that the packets can hold executable code along with their data payload. In other words, it can be programmed. Possible advantages of such a feature include faster expansion of further services, the capability to adapt services for different applications and a capability to automate services and management configurations.

As a result, the advantages for both policy management and active network technology are equal. "On one hand, active network is a kind of enabling technology for policy enforcement; on the other hand, policy Management also provides the management of active networks themselves"[119]. After presenting an active network approach into grid management, a modern active grid management architecture appears offering methods that will dynamically fit grid network parts to different grid applications and helps to organize the grid system itself. Figure (2.11) shows the overall policy active grid management architecture. The approach to add programmability to grid

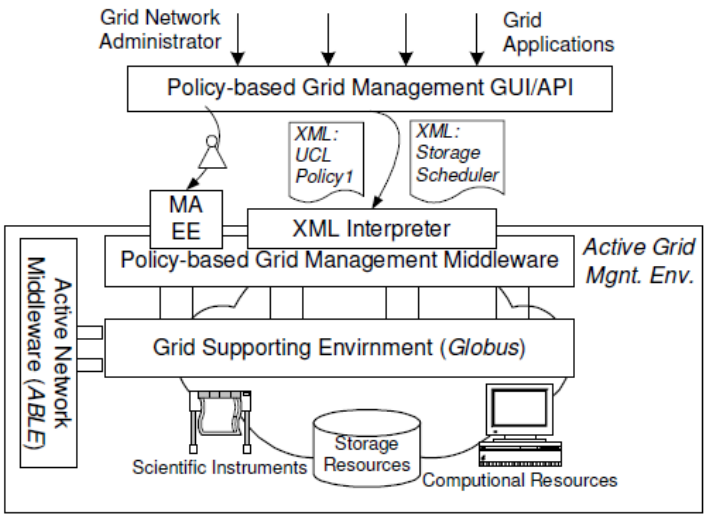

Figure 2.11: Grid System Architecture [119]

management is to expand the broadly used grid supporting tool (Globus) by the means

of middleware. Together active network middleware and policy grid management middleware may be used by grid supporting environment to ease the operational so as to obtain better handling and management of multi grid resources such as, computing resources, massive storage resources and special scientific instruments [119]. The core of the active network is the active node, which is based on the ABLE. ABLE is "an active network architecture that primarily addresses the network management challenges. It's main component is the active engine that is attached to any IP router or massive storage or computational resources to form an active node" [78]. Another definition for ABLE is "a complete software environment dedicated to deploy active routers and services inside the network. It provides persistent active routers, which are able to handle different applications and various data stream (audio, video...) at the same time" [57].
Policy Enforcement Points (PEP) represents the end point, where the policy is finally applied. To achieve this application, a transport protocol should be presented for the purpose of communication between Policy Decision Point (PDP) and PEP, so that the user can send policy regulations or configuration data to the end point(s), or read configuration and get information from the device. Active network technology has become the most popular way to achieve policy enforcement [119].

2.14 Policies for Data Movement

Feng, Cui, Wasson and Humphrey in [44] mentione a new system that deals with policies that control data movements. This system is divided into two parts; the first one is called the MyPolMan which is responsible for organizing, uploading and distributing policies over the PEPs. The second part is called the .NETGridFTP a data transfer application protocol that enforces these policies. Figure 2.12 shows how such a system organizes the data movements over the grid.
MyPolMan is concerned with policy management in grid environments. This allows administrators (or users) to distribute policies to a repository that can then be recouped by other system entities as required. As MyPolMan is used to handle many kinds of policy (not only data policies), many policy construction tools may be used to build up

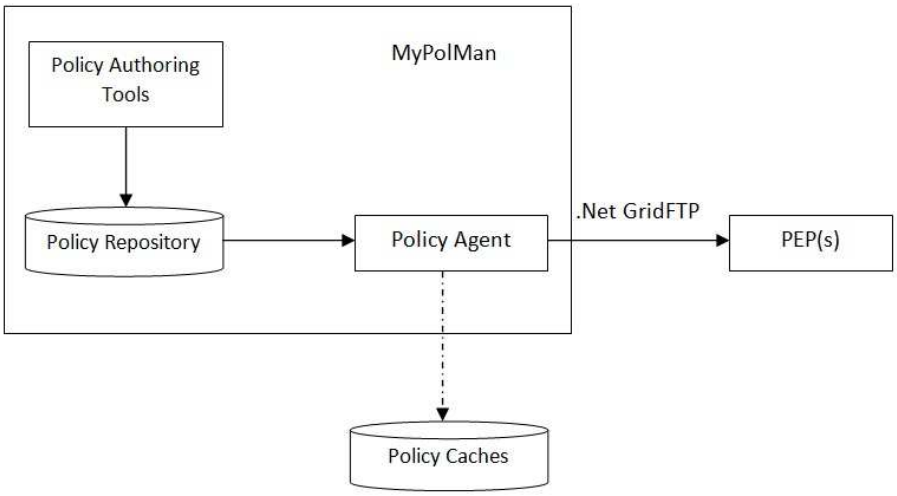

Figure 2.12: Data Movements over the Grid System [44]

the policies that are available in MyPolMan. The resulting policy from the MyPolMan, expressed in XML According to the WS-Policy [22], which has a unique identifier (PolicyName) and a group of policy attributes. These attributes include the policy author, applicable scope, valid time and current status, in addition to other parameters. After the policies have been uploaded to MyPolMan service, services called policy agents are used to assist policy distribution between the policy service and policy consumers. As the Application Scope field is used to indicate the services to which agent this policy applies, the policy can then be moved to the policy agent who caches it on the local repository system. This caching enhances performance and enables many co-located PEPs to make use of the same policy agent. Any changes made by the administrator to the policy will cause these steps to be repeated.

Although the GridFTP protocol offers security for Grid data movements, there is no clear policy support to apply resource employment policies that are saved in the policy server. The .NET GridFTP, which is an implementation of GridFTP on the Microsoft .NET platform executes Grid data transfers bidirectional between Windows machines and enforce resource employment policies that are saved in MyPolMan.

2.15 Grid Computing Challenges

2.15.1 Security Challenges

One of the important issues that most researches try to solve it is how to keep distributed recourses from being access by unauthorized users and at the same time allow the sharing of resources and accountability for resource handling. David G. Rosado and others define grid systems in relate to security aspects as following: "Systems based on Grid Computing are a type of systems that have clear differentiating features of which security is an extremely important aspect" [99]. Grid environments have extraordinary characteristics that cause them to be distinguished from other systems, such as resources pool, user population, and the fact that the sets of processes running on multi and various sites are possibly huge and dynamic. All of these characteristic must be taken into consideration all through the overall development lifecycle. In addition, we should consider that processes may communicate by a range of methods such as multicast or unicast, and several authentication and authorization methods can exist in an individual job in relation to the local security policies of the sites involved. Individual users may be belong to various local name spaces, accounts and credentials on various sites [53, 98]. Every resource applies a local security policy that may result in the refusal of requests for utilizing of its resources. This presents problems for both resource requestors and providers who want to share resources in the lack of global security policies in grid systems. Because of the fact that there are a lot of elements like users and resources contributing on the grid, security has become a critical aspect in checking the element trying to use a service (authentication), and verifying either this element is allowed or not to use the service (authorization). Securing the grid is vital to give confidence to both resource providers and users to join the grid. Globus GSI (Grid Security Infrastructure) is an environment that cares about the elements security in the grid by using public keys for authentication [24]. Security in distributed systems can be subdivided as following:

- **Identity and Authentication**

  Public Key Infrastructure (PKI) [24] along with the Transport Layer Protocol

(TLS) [23] is a common way to resolve problems including overall identity and authentication in grid systems. Internet2 sites [23] use either a user's authentication process among multiple systems or organizations (federated identity) or attribute server solutions. In Kerberos identities [103] sites try to combine these identities with grid ones, which makes user credential organizing more manageable. As authentication is the first action to utilize and access resources, grid authentication should act with both user and resource site authentication conditions.

- **Secure Communication**

  Secure connections are typically managed by TLS [38]. The Globus Toolkit GSI library based on top of SSL controls the communication in distributed computing. These protocols must be expanded to handle all secure group communication aspects for grid applications.

- **Authorization**

  Authorization demands vary with each application [111]. The complication occurs from the naming of the users and resources who organize the authorization policy. David G. Rosado and others in [37] describe the authorization in grid systems as "The core problem with authorization in a grid setting is how to handle the overlay of policies and other assertions from multiple administrative domains (user policy, VO specific policy, operational procedures, and site-local policy)". A high-quality access policy should have the transparency feature to grid users and be simple for administrators to organize and keep up. For any site provides resources, the authorization requirements should have registration for resource use, accountability and quality service assurance. One more essential thing is to name users in a meaningful way for both the resource site and throughout the grid.

- **Privacy**

  As a result of shared resources in grid distributed systems, insecurity and privacy abuse are the main barrier to growth of distributed systems applications. Most of

recent research is working on this matter to protect the grid community and offer assurance of privacy.

As an example of the grid privacy, let's assume there are two domains in the United Kingdom (UK), one domain in the USA and one in China. The domains in the UK are allowed to share their data and application software's with the USA domain but not with the China one. In the other side, the USA domain is allowed to share its data with all domains. As we can see, there is a need for a tool to control the moving of data and application software's between these domains preventing the UK data and application software's from being used in the China domain.

2.15.2 Accounting

Accounting is a fundamental necessity of businesses. Many ways have been presented to solve this issue but they are yet in the early stages. One idea is to sign users at a local site which is checked or treated as it would be for any user. This makes users' use of several resources across the grid easier, because the cost for such use will go back to the local sites where users were signed in the beginning. But this idea and many more ides are still an area for basic research, as many efforts are ongoing to develop these ideas.

2.15.3 Resource Brokering

Grid resources originally are heterogeneous in architectures, operating systems, speed, polices, data and application software, along with the geographically distributed attribute. For these reasons, there is a need to locate the available and suitable resources on the grid. Another problem of resource allocation is the lack of prober information about the status of resources [59], because of the dynamically nature of grid resources. According to that resources, information should be updated frequently to give accurate information about their status.

2.15.4 Job Description

The accurate description for grid jobs considers an essential requirement for resource management and scheduling within a grid environment. Various languages such as Globus Resource Specification Language, European Data Grid JDL and Job Submission Description Language (JSDL) have been introduced for job description. But none of these languages can deal with applications or events.

2.16 Summary

This chapter surveyed the background and related work on all the major research issues covered in this thesis. It has illustrated distributed systems' attributes and architectures. This was followed by a survey of grid computing, as well as the categories into which it is divided. The following section described the structural design of grid computing followed by a review of the mobility in grid computing. Section four presented grid resource broker, including tasks and schemes. Section five described mobility in grid computing. Section seven introduced type of security solutions introduced in grid computing. Section eight presented policy definition and category. Section nine described grid technology infrastructure. Section ten introduced languages for describing grid jobs. Section eleven illustrated the dynamic policy management framework that has been introduced by Globus toolkits . Section twelve presented active network technology used to support grid system management. Section thirteen described the recent used policies for data movements. And the final section, fourteen, presented the grid computing challenges. The reader needs to bear in mind the following sections: Resource broker, Mobility, grid policy, Active network and Job description language.

Chapter 3

The Architectural Model

3.1 Introduction

Grids depend on enhanced software that guarantees seamless communication between node components. It uses an effective mechanism which determines the suitable policy(s) that should be applied to achieve the best way to utilize resources in a way that guarantee privacy and security for both grid users and grid resources.

The grid infrastructure allows contribution and sharing at the level of a Virtual Resource (VR). The VR varies from a single machine, a group of machines or a virtual partition on the same machine. Each grid institute has many VRs can participate with other contributors in Virtual Organizations (VO) [113]. A virtual Organization (VO) seeks to supply authorization methods in which policies are specially described to that specific VO. In addition to VO special policies, local site special policies also exist [31]. Due to the heterogeneous nature of grids, a conflict between these (VOs) might take place in the security policy framework [66].

The problem with most of these traditional policy frameworks is that concentrating at the policy management inside the VO rather than the management between multiple VOs [105],[112],[116]. External users, who want to utilize resources in the grid need guarantees for their jobs and data, most recent policy frameworks do not take that into consideration. To data, not enough attention has been paid to policies that deal with

its movement within the grid; most existing grid systems support only limited types of policies (e.g. CPU resources). A few frameworks consider enforcing data policies in their architecture [44], [60],[81]. Our new framework is different from other policy management frameworks in that it takes into the account the external user preferences along with enforcing policies for data movements within the grid.

This chapter presents an overview of our new architecture, and considers one of our major contributions in this thesis. It is proposes an extension to the framework in [121] to be able to provide the features of supporting the grid user preferences along with enforcing policies for data movements and resource mobility feature for a single virtual organization and multiple virtual organizations; within different domains under different administrators. The framework in the second section has been published in the Communications in Computer and Information Science Conference (CoNeCo2011) [15] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110]. The next section will present the components of our grid architecture. Section four and five detail the interaction between user and grid environments, which constitutes the grid portal, and the interactions between the institute policy agents and the grid main policy agent in section six. The framework which will be introduced in the last section has been published in the Risk and Security of Internet and Systems Conference (CRiSIS), IEEE Computer Society, 2011 [14] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

3.2 Framework for Policy Management

Our new framework is different from other policy management frameworks in that it takes into the account the external User Policy (UP) along with enforcing policies for data movements within the Grid. The following two subsections describe how our framework within a single VO institute and also Multiple VO institutes.

3.2.1 A Single Virtual Organization

Our framework consists of three agents: a Policy Agent (PA), a Policy Management Agent (PMA) and a Grid Information Agent (GIA). Figure (3.1) shows the framework for a single VO. Each Virtual Organization (VO) should have at least one policy agent that has the ability to access the policy repository. For PAs of the same security policy framework, there should be a PA leader that coordinates other PAs in the cluster and at the same time performs a homogenous and a heterogeneous policy management across different policy frameworks. This agent is called (PMA). The administrator of the institute specifies and stores policies at the Policy Management Agent (PMA). The (PMA) can be considered as a combination of the policy management tool and policy repository. The (GIA) which is owned by the grid administrator is responsible for providing PMAs with the necessary information that is needed to perform the heterogeneous policy management, if it is necessary, across different policy frameworks.

Grid services are a field of web-services and for this reason the (PMA) would be a web service that publishes the set of services that can provide for an institute into the grid registry. For remote access, the (PMA) supports a SOAP/HTTP protocol binding in order to swap documents easily over SOAP. The policy documents conforming to the specification of the common information model are encoded in XML [113].

From Figure (3.1), it can be seen that three main features have been added to the policy management framework mentioned in [121]; First, it enforces the data policies management by using NETGridFTP protocol [44]. Although the GridFTP protocol offers security for grid data movements, there is no clear policy support to apply the resource employment policies that are saved in the policy management agent. The NETGridFTP, which is an implementation of GridFTP on the Microsoft. NET framework, can execute grid data transfers bidirectional between windows machines and can enforce the resource employment policies that are saved in the PMA.

The second feature is using the Active Network Middleware to connect the Policy Decision Points (PDPs) with the Policy Enforcement Points (PEPs). The advantages for both policy management and active network technology are equal. “On one hand, active network is a kind of enabling technology for policy enforcement; on the other

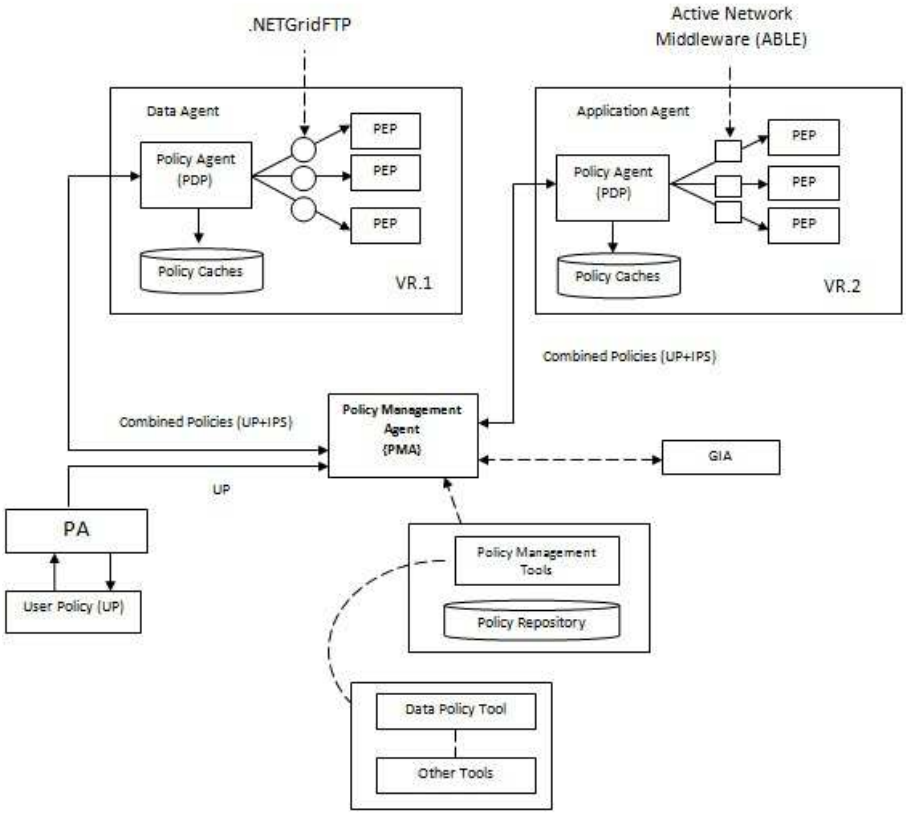

Figure 3.1: Single Vrtuial Organization Policy Management Framework

hand, policy Management also provides the manage-ment of active networks themselves" [119]. This approach to add programmability to grid management is to expand the broadly used grid supporting tool (Globus) by means of suitable middleware. Together, active network middleware and policy grid management middleware may be used by the grid supporting environment to ease the operation so that they can obtain better handling and management of multi-grid resources such as computing resources, massive storage resources and special scientific instruments [119].The core of the active network is the active node, which is based on ABLE (defined before in chapter 2 on page 47).

Policy Enforcement Points (PEP) represent the end points where the policy is finally applied. To achieve this application, a transport protocol is presented for the purpose of

communication between Policy Decision Point (PDP) and PEP so that the user can send policy regulations or configuration data to the end point(s), or can read configuration and get information from the device. Active network technology has become the most popular way to achieve policy enforcement [119]. Another advantage in our architecture is taking the external User Policy (UP) into the account when making the final conflict decisions. When the users have completed describing their jobs and adding their own policies to that description, they send the descriptions along with the policies to the resource broker who forwards the policies to the policy management agent. Later, the (PMA) enforces these policies by its turn in its final discussions along with other policies (grid and resources policies).

3.2.1.1 Policy Tools

Policy tools offer methods so as to take policies from other management stations. These tools help administrator to create new policies for the environment or view or edit any existing policies in the policy repository. The policies can be written using different level languages. In this case the tools have to do a translation from these languages to ones that can be understood by PDP.

Policy tools primarily consist of Policy Receiving Module and Policy Editor (GUI). The Policy Receiving Module is represented by a fixed representatives or agents. This agent is responsible for receiving XML policies transferred from higher management stations. The Policy Editor GUI supplies an easy way for administrator to input some simple information to produce XML based policy automatically and save it into the policy repository [119].

3.2.1.2 The Policy Repository

The policy repository is a place where the policies are stored. After describing and authorizing them by the policy management tool, polices are stored in this place waiting to be retrieved either from the policy decision point or the policy tools.

3.2.1.3 Policy Decision Point (PDP)

PDP is the entity that is responsible for retrieving policies from the repository. After that the PDP reads policies and pase them with the help of XML parser, i.e. it verifies the authority of user policy to examine the privileges for services approved to any grid user. If the PDP decides that the user has the right to use these services, PDP looks at the conditions of the policy and asks if there are sufficient resources to apply this policy, then it decides when this policy should be applied [119].

3.2.1.4 Policy Enforcement Point (PEP)

This point represents the end point, where the policy is finally enforced. To achieve this enforcement, a transport protocol is presented for the purpose of communication between PDP and PEP, so that the user can transmit policy conditions or configurations data to the target, or retrieve these configurations and information from the device. Besides COPS and SOAP protocols, active network technology has become the most popular way in use used for policy enforcement [119].

3.2.2 Multiple Virtual Organizations

Virtual organisation (VO) can be defined as a short-term network of organizations and entities that get together fast to make use of fast shifting and sharing chances for the period of the chance exists. It provides organizations the flexibility to exploit and co-operation with partners [71].

Trust can be refers to "mechanisms to verify that the source of information is really who the source claims to be" [16], and how secure to share resources (application software and CPU process) and private data without any concern form being stolen or compromised.

Figure (3.2) shows our framework for Multiple VO. Allowing PEPs (e.g: external users) to obtain policy instructions from subjects outside their physical institute exposes them to security defencelessness. To avoid this issue, each PEP should be remaining only under the administrative control of the policy management agent (PMA)

in its physical institute. Our framework deploys PAs to divide VOs into virtual clusters according to their security policy framework.

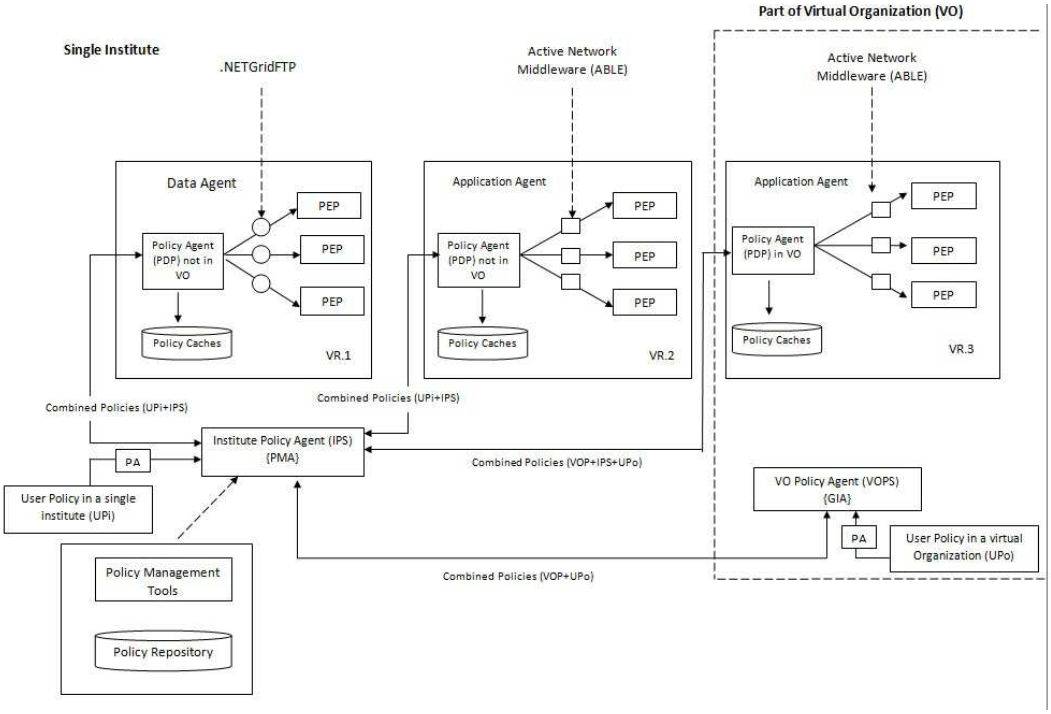

Figure 3.2: Multiple Vrtuial Organization Policy Management Framework

When authorized grid users send their jobs to the resource broker, the later retrieves information about the available resources in the grid from the Grid Information Services (GIS). Later, the resource broker sends this information along with the related policies (Users policies) to the grid policy agent or to the VO policy agent to make the policy decisions. The VO policy agent forwards the users' policies along with the VO policy to the PMA which is responsible for the target resource to make the final decisions. PMA checks if the requests are situated in a single virtual cluster or multiple virtual cluster. If the requests are situated in the same virtual cluster, a homogenous conflict analysis takes place without any need to retrieve any information from the GIA.

Where PMA receives a request from one of the PAs asking for a service that is situated in multiple virtual clusters, a heterogeneous policy management mechanism takes place. The heterogeneous conflict analysis can take place in two cases; either the PMA

receives a request from one of the PAs that has a framework different from that in the PMA or if the User Policy framework is different from that in the PMA. The heterogeneous conflict analysis mechanism depends on converting the policies of the target service into the policy model of the PMA. The PMA asks the GIA for the account maps and the policy schema maps of the target services. At the same time, the PMA asks the service requester for its authorization policies through the PA of the service. When the PMA receives this information, it starts the policies conversion to the PMA policy model. First it applies the Account Mapping (as explained in the literature review chapter) to make it possible for users, whether trusted or not, to access services in a remote VO. Here, a map mechanism should be applied to map those users to local accounts and later to perform the policy mapping to generate an inter-schema map which maps the schema of the policy model of the service's VO to that of the requester's VO. In other words, a map mechanism is used to convert policies between heterogeneous policy models to a one that can be understood by the PMA. Finally, the PMA applies the conflict analysis mechanism on the policies of all target services to find suitable permissions for the requester of service.

3.3 Architecture Structure and Components

Figure (3.3) shows our proposed architecture, it applies Client/Server architecture because it is the most favorable type in distributed systems and heterogeneous environments [35]. Client/Server is a network computational architecture and includes clients and servers that operate on the proper software and hardware for their jobs. There are two forms of client/server architecture: two-tier and three- or multitier. Our architecture employs the three-tier model which consists of the client (grid portal) as the first tier, the resource broker as second tier and grid nodes as third tier.

Our grid architecture composes of a grid portal, a resource broker with its units and nodes. The following describes the functions for each one of them.

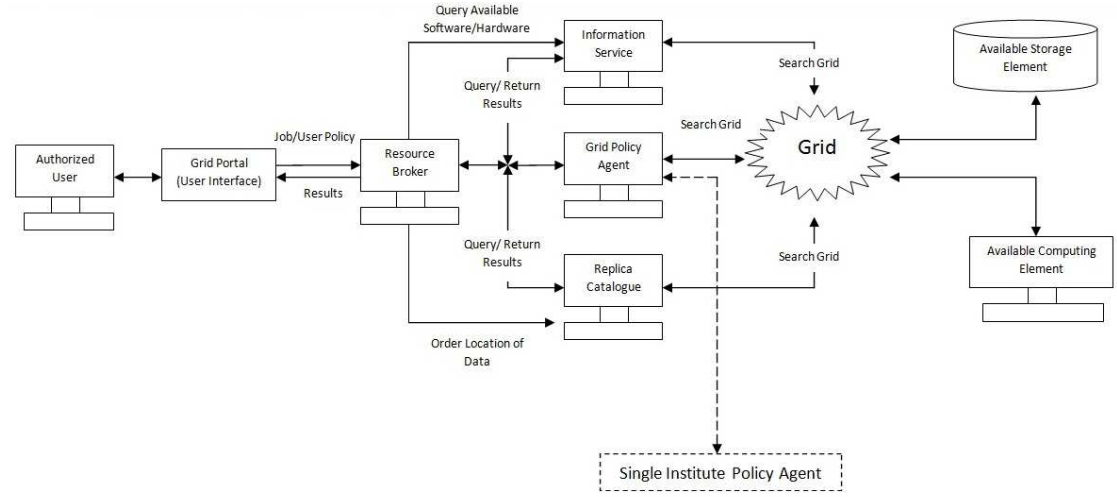

Figure 3.3: Grid Architecture

3.4 Grid Portal

A grid portal or grid interface is a virtual computing resource performing an interface on behalf of grid users to approach the grid; the portal is the efficient gateway to the grid, as shown in Figure (3.4). A portal has many features such a simple interface by which facilitates the description of grid applications and job necessities. Web and grid portals support identical services to users: for example, web browsers offer a single interface that can be used to access internet resources, where grid portals are used to illustrate and send job or applications (or both) to be accomplished by grid resources.

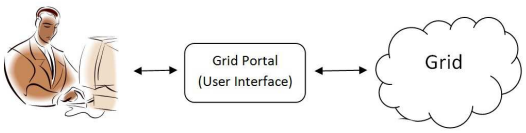

Figure 3.4: Grid Portal

3.4.1 Grid Application Requirements

A grid application is a group of jobs require being executed by using grid resources. These applications have requirements like the names of all integrated jobs, the relations

between these jobs and the single job requirements. These application requirements must be taken into account by the resource broker, which must then choose the best resources that fit those requirements so as to accomplish the job. This stage is essential, as it will affect the stages that specify the nodes on which the application will be implemented and how events will be treated.

3.4.2 Job Requirements

Users need to specify their job's requirements. These comprise of job's (ID) or identification, resource specifications (CPU architecture, physical memory and disk space), data staging, software applications and policy. This stage is as vital as the previous one, for the same reasons.

3.5 Grid Node

Node is a shared unit offering particular facilities. These units are the essential elements in any grid. Every node has to communicate with a broker in addition to other nodes in the grid system to enable other nodes to perform their jobs that were requested by resource broker to decrease the resource broker bottleneck problem.

To achieve this communication, each node has to include middleware application software. This communication should be made by the use of network resources. In our model the resource broker does not organize this network. Each node has the ability to execute a job, because each one of them has its own application software, policy and data.

Every node transmits its heartbeat to the resource broker at regular intervals. This heartbeat "pulse" includes a timestamp, node name or ID and other optional information. The node allows the resource broker to inquire its resources and grid jobs status and control the grid jobs (for example: stopping the job). It also transmits any event for handling to the resource broker throughout the execution.

3.6 Resource Broker

The resource broker presented in this research is a modified one from that mentioned in Section 2.5. Our resource broker is based on the mobility framework and isolate the user from the grid's middleware. All of this helps in automating the operation from the point of receiving user's resource through the job (or application) execution on the appropriate resources till the submission of results. In other words, the resource broker is connected to all grid elements.

3.6.1 Resource Broker Architecture

The resource broker accepts a job or application requirements from the portal and looks for appropriate resources that can fulfill these requirements. First it asks for all information about the available resources from the information service and about the data information held in the replica catalogue. It chooses the resources that can meet the job (or application) requirements and asks the grid policy agent about the policies for those resources. According to findings, the resource broker's architecture consists of three indices: the information service, the replica catalogue and the grid policy agent. These may configure by the grid administrator, either to gather all information about the resources, or to include only the addresses of information services sited on the resource.

3.6.1.1 Information Service

Information service is a crucial element in grid computing. It is a directory service holding information about all the resources in the grid and the entire grid activated jobs running on those resources. This information can be static for the hardware conditions and the operating system, or dynamic information related to the resources available time, disk space, the job presently running, application software, and policies. The resource broker communicates both the information service to ask for this information and the resources in order to advertise their information.

3.6.1.2 Replica Catalogue

It is also an important component for the grid, because it presents information which helps in accessing the stored data in the grid. It determines the place of data in the grid, updates data resources and maps logical file names to the actual physical places on grid resources. A resource broker communicates with a replica catalogue to ask for information about data location and the access control needed to use this data.

3.6.2 Grid Policy Agent

Our grid policy agent contains all the policies information about all resources in the grid. Each institute should have at least one policy agent that has the ability to access the policy repository or policy information for that institute. All policy agents (PA) in all domains in the grid are registered with the grid policy agent, and should send their policy information (e.g. policy framework) or any changes or updated data about their policies to the grid policy agent. Grid services are an area of web-services; for this reason the policy agent is a web-service that issues the group of services that it can support for an institute into the grid registry. For remote access, the policy agent supports the SOAP/HTTP protocol so as to exchange documents easily over SOAP. Policy documents meeting the requirements of the specification of the common information model are encoded in XML [113].

The grid administrator can specify the policies for units participated in the grid, but it does not have any policy agents that can directly use it. As an alternative, a grid policy agent operates as a proxy for the policy agents that run at each of the different institutes.

Figure (3.5) shows the architecture of a single institute policy agent. Each grid institute has many Virtual Resources (VRs) that are accepted to participate with other contributors in the grid. The grid administrator defines policies using various management tools in the policy agent and stores those policies in the policy repository. As a result, the institite policy agent can be considered as a combination of the policy management tools and policy repository [113].

The main job for the institute policy agent is to combine the policies from the institute administrator, the policies from the global grid and grid user's policies in order to obtain the efficient set of policies for resources belonging to that institute. The efficient set of policies is the one applied by the policy agents attached to each resource assigned to that institute in the grid.

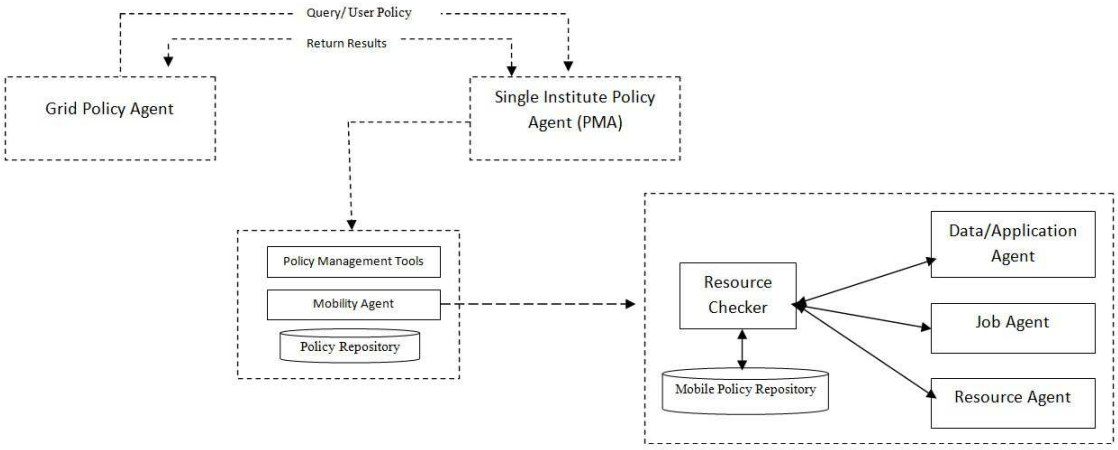

Figure 3.5: Mobile Agent Architecture

In many cases resource broker find itself in a situation that has to reject some jobs because the required resources may not be found. These are some of these situations, in short:

- The resource needed to fit the job requirements is busy at this time.
- Resource that fits the job hardware requirement does not own the needed application software.
- The resource that fits the job hardware and application software requirement does not have the required data.

As a result, the Mobility has created a new environment that can solve these rejected job cases. The requirements of privacy and security to apply the mobility in secure grid environment are one of the important demands for both grid user and grid resources. In our model mobile policy agent play a significant role in achieving these requirements. Figure (3.5) shows the architecture of our mobile policy agent and its components. The

following describe each of one of them.

In order to apply the mobility, the policies for the elements in Figure (3.6) are checked before any dynamic migration can take place. In our model mobile policy agent plays (Figure (3.5)) a significant role to achieve these requirements. The mobile policy agent checks the policies for each element in the three levels (Figure 3.6). Each element in these levels is consider as a Policy Decision Point (PDP) where the policy decision is taken place and forward this decision to its related agent. If the policies at (level 1) have given the green light for the migration, the mobile policy agent checks the policy for the target domain (Level 2) to see if that domain is allowed to have all (or any) of the elements in (Level 1). If so, the next step should be checking the node which is going to be the new host for the elements in (Level 1). By checking the policies for both elements in (Level 1) and node's policy in (Level 3) mobile policy agent takes the decision if that node is allowed to have the immigrant element in (Level 1) or not.

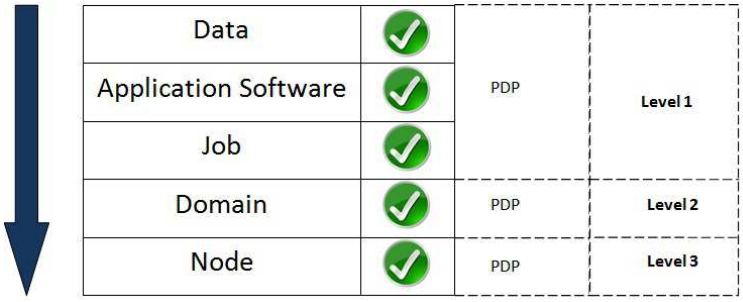

Figure 3.6: Policy Levels

3.6.2.1 Data/Application Software Agent

This agent is responsible for the data and application software migration. Our grid architecture allows application software and/or data to migrate from one node to another in the grid environment so as to adapt the resources needed to fit the job requirements. If the resource fits the job hardware conditions and the availability time, but does not have the application software or data needed for the job(s), the resource broker will search the grid for the nodes that have this data/application software, by inquiring the information service and replica catalogue and put these nodes in a new list. Then it

will check each one of these nodes, one by one, by asking the mobile policy agent to determine whether or not the data/application's software policy in these nodes allows their movements (or having a copy from this data or application software). The Data/Application Software Agent will check the policies for the nodes that contain the required data or application software and return the results to the resource broker. If one of the nodes does support the mobility feature for data/application software, the resource broker will migrate or copy appropriately and send it to the resource that meets the job hardware and time requirements along with its policy. If all the nodes' policies do not support the data/application software mobility, the broker will tell the user that the grid cannot execute the job.

3.6.2.2 Job Agent

This agent is responsible for checking the grid users' policies. Our grid architecture in Figure (3.5) allows the job and its execution state (i.e. the context of execution) to migrate from one resource to another and restart on the new one in order to fit the job conditions and requirements. If the resource that fits the job hardware requirements is busy at the time needed, our resource broker will vacate this resource by migrating the currently running job in that resource to other resources (if they are presented and have the job requirements). This can be done by looking for jobs running on this required resource and obtain details of their requirements from the information service and the replica catalogue. If the job requirements can be satisfied using other resources, the resource broker will ask the mobile policy agent if the currently running job(s) is allowed to be migrated to another resources. The Job Agent and the Resource Agent in the mobile policy agent will check whether or not the grid user's policy and the new resource(s) policies allow migrating the running job to the new resources and returning the results to the resource broker. If the policy allows this kind of migration, then the resource broker will migrate these jobs to the new resource(s) and send the new job to the vacated resource which will fulfill its requirements.

3.6.2.3 Resource Agent

This agent checks whether or not the resources' policies allow the migration for jobs, data and application software between various resources. Our grid architecture allows jobs, data and application software to migrate from one node to another in the grid environment in order to adapt the resources required to meet the job requirements. In the case the resource that meets the job requirements is currently busy and there is a need to migrate the currently running job(s) to other resource that can meet the running job requirements, or there is a need for a data or application software migration. In both cases, the resource broker will ask the mobile policy agent to check the policy aspect in these situations. The Resource Agent in the mobile policy server will determine whether or not the current resource's policy allows the job migration from its node to the destination resource, or if the destination resource can accept jobs from the original resource. In both cases, it will inform the resource broker about the results. In the case of data/software migration the resource agent in the mobile policy agent will determine if the addition or migrating of data/application software policies are allowed in the current resource and the destination resources. If they do not, the broker will tell the user that the grid cannot execute the job. If they do, the broker will apply the migration between those resources.

3.6.2.4 Resource Checker

As soon as the mobile policy agent makes its decisions about any possible migration(s) either for jobs, data or application's software, it stores indexes for these decisions using the resource checker and stores these indexes in the policy repository prior to submitting the decision's results to the resource broker. The aim of these indexes is to track any changes or updates in the target policy(s) and inform the resource broker about them. Also it helps in enhancing the mobile policy agent performance and throughputs by returning to these indexes for any new requests from the resource broker instead of going for the whole checking operation again.

3.6.3 Resource Broker and Grid Policy Agent Functionality

The main tasks for the resource broker is establish job start times, discovering resources, choosing the best resource(s) that can meet the job requirements, using the migration features (if required for jobs, data and application software), booking resources, sending jobs input files to the resources, monitoring and handling jobs and transferring results back to user. Resource brokers acquire information about all the needed grid resources from replica catalogues, information services and grid policy agent. All these progressions are hidden to the user, who sees the grid as a single large and powerful computer.

In the case of grid application, the resource broker will verify the requirements for each job in this application and locate the resources that can perform each of them. If there is an unavailability in one of these resources, the resource broker will tell the user that the grid will not be able to execute this application.

3.6.3.1 Job Start Time (JST) establishment

As soon as the resource broker accepts a job from a portal it will determine the job's start time using the algorithm in Figure (3.7).

1. **Start and initialization.** The first action is to rest all the variables inside this algorithm to zero. For single Job the resource broker will reset the JST to 0, which will launch the job execution right away, and will then discover the resources.

3.6.3.2 Resource Discovery

After the resource broker starts the job execution task, it will contact the information service to ask about available resources' information (hardware and software), the replica catalogue to inquiry about data that can meet the jobs requirements and the grid policy agent to query about the allowed actions for both the grid user and grid resources. The following describes the algorithm for resource discovery.

2. **Checking Hardware Specification**. The resource broker will be given some or all of the hardware specifications, that include; CPU speed, memory and disk

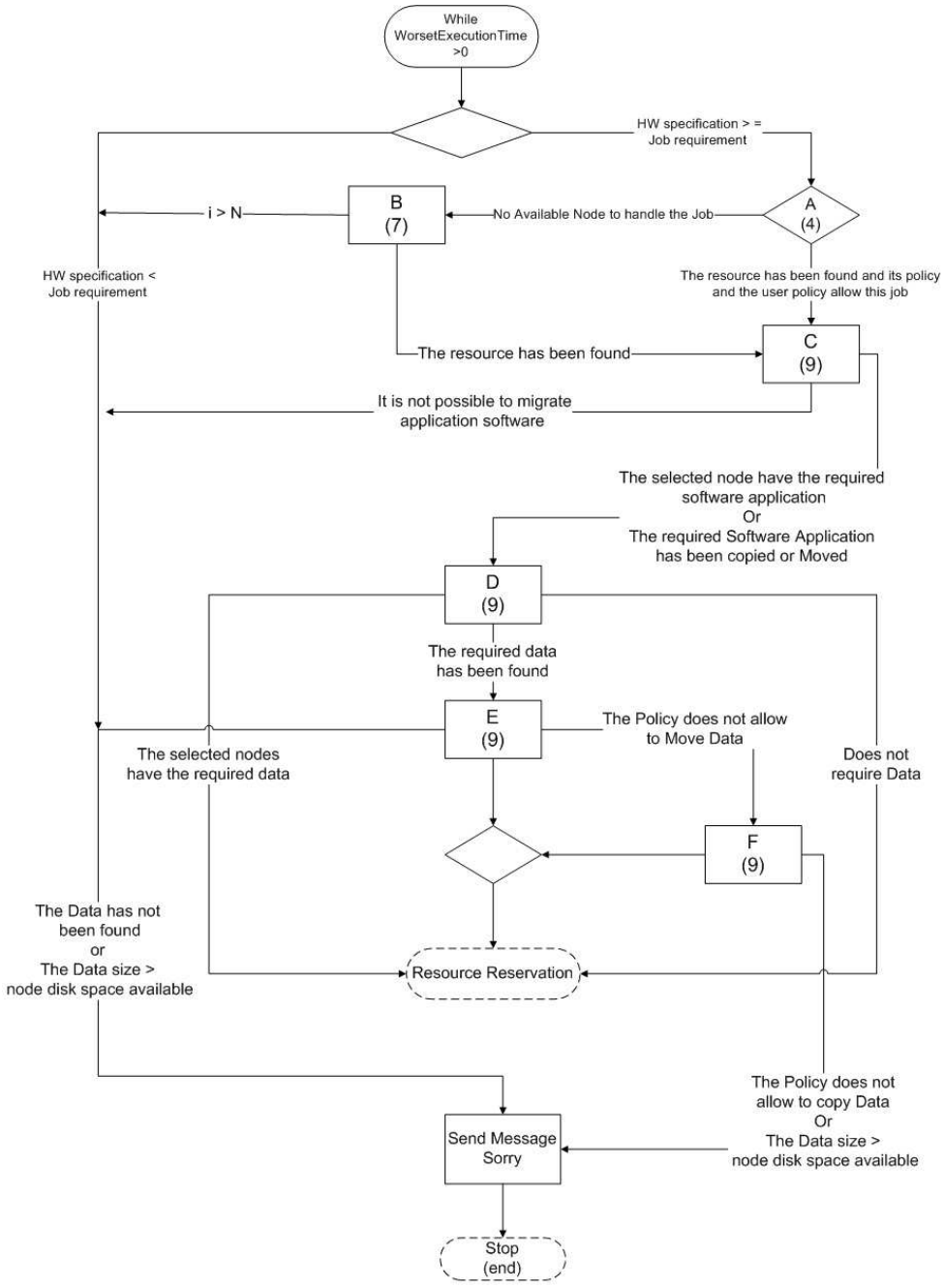

Figure 3.7: Resource Discovery and Mobile Policy Algorithm

space.

The resource broker will evaluate if the given hardware specifications from the grid user are equivalent or larger than that saved in the information service and

place the result into a new list in order to locate the suitable resources for the job. If one or more of the required hardware specifications is not defined or identified from the user side, the resource broker will put any value. If the resource broker does not locate any of the needed hardware specifications, it will transmit a message to the user saying that the grid does not have the needed resource. Otherwise the resource broker will execute the next step.

3. **Checking Time**. The resource broker will have the following times:

   - Worse Case (WorseCase) which is the maximum time that the job can be held until it is executed. Several issues such as heterogeneous environments, communication delays and data transfer speeds cause difficulties for the determination of job execution times.
   - Job Start Time (JST) (generated from the system).

4. **Choosing Suitable Resources**. The resource broker will check the policy for both the resource and the user. If the resource allows executing this job(s) using its resources and the user policy does not prevent executing the user job(s) on this resource, the resource broker will perform the next step. If it does, the resource broker will look for another resource(s) to execute this job(s). Figure (3.8) illustrates this step.

5. **Application Software**. If the user's job needs specific application software to fulfil the job requirement, the user will provide the resource broker possibly will be given Application Software Name. Then the resource broker will compare the needed application software. If the resource broker does not locate the needed application software, it will execute to the application software migration step. Otherwise the resource broker will go to the following step (Step 6). Figure (3.9) illustrates this step.

6. **Data**. If the user's job needs specific data to fulfil the job requirement, the user will provide the resource broker with a possible data name. The resource broker will compare the needed data name with one on the listed resource; if it does not

Figure 3.8: Resource Discovery Step (4)

find the needed data name, it will perform the data migration step; otherwise it will execute the resource reservation step. Figure (3.10) explains this step.

3.6.3.3 Mobility

Mobility is the ability to move or migrate jobs, data and application software among grid resources according to their policies. Mobility facilitates the accomplishment of requirements for grid applications, as well as grid users. It also assists grid evolution, improves performance of operating applications by migrating data to the execution host and therefore reduces the communication consumption, solves the load balancing and reduces latency and bandwidth consumption when applications are migrated to locally interact with remote data [87].

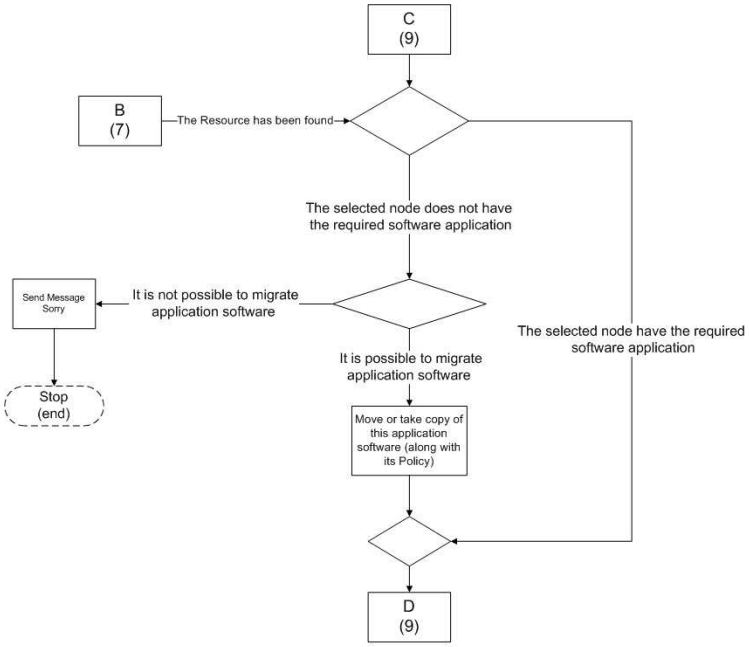

Figure 3.9: Application software Mobility Steps (5 and 9)

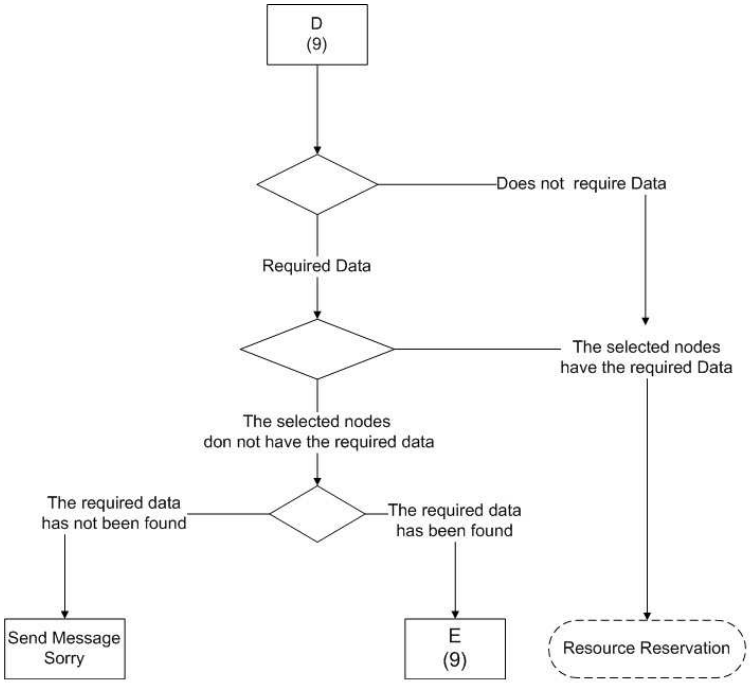

Figure 3.10: Data Checking Step (6)

7. **Job Migration**

   Our grid architecture allows the job and its execution state (i.e. the context of execution) to migrate from one resource to another and restart on the new one in order to fit the job conditions and requirement. As illustrated in the following algorithm, if the resource that fits the job hardware requirements is busy at the time needed, our resource broker will vacate this resource by migrating the currently running job to other resources (if they are present and have the job requirements). The resource broker will ask the mobile policy agent if the user of currently running job(s) allows migrating this job to another resource(s) and asks if the policy of the target node is willing to accept the migrated job and continue using its resources. The Job Agent in the mobile policy Agent will check whether or not the grid user's policy allows migrating the running job to the new resource(s) and returning the results to the resource broker. If the policy allows this kind of migration, then the resource broker will go to the next step. If not, the resource broker will check the possibility to migrate the following jobs in the resource broker list. Figure (3.11) illustrates this process.

8. **Resource migration**

   In this case the user policy allows the migration of the user's job from one node to another, the next step is to check whether the resource which hosts the currently running job allows migrating the running jobs to another resource. The resource broker will ask the resource agent in the grid mobile agent to determine whether or not the current resource's policy allows the job migration from its node to the destination resource. If it does, the resource broker will calculate the rest of the execution time and check the job hardware requirement and apply the migration to the suitable resource. If it does not, the resource broker will check the possibility to migrate the following jobs in the resource broker list. Figure (3.11) shows this process.

9. **Data/Application Software Migration**

   If the resource and the user allow the migration of their jobs from one node to

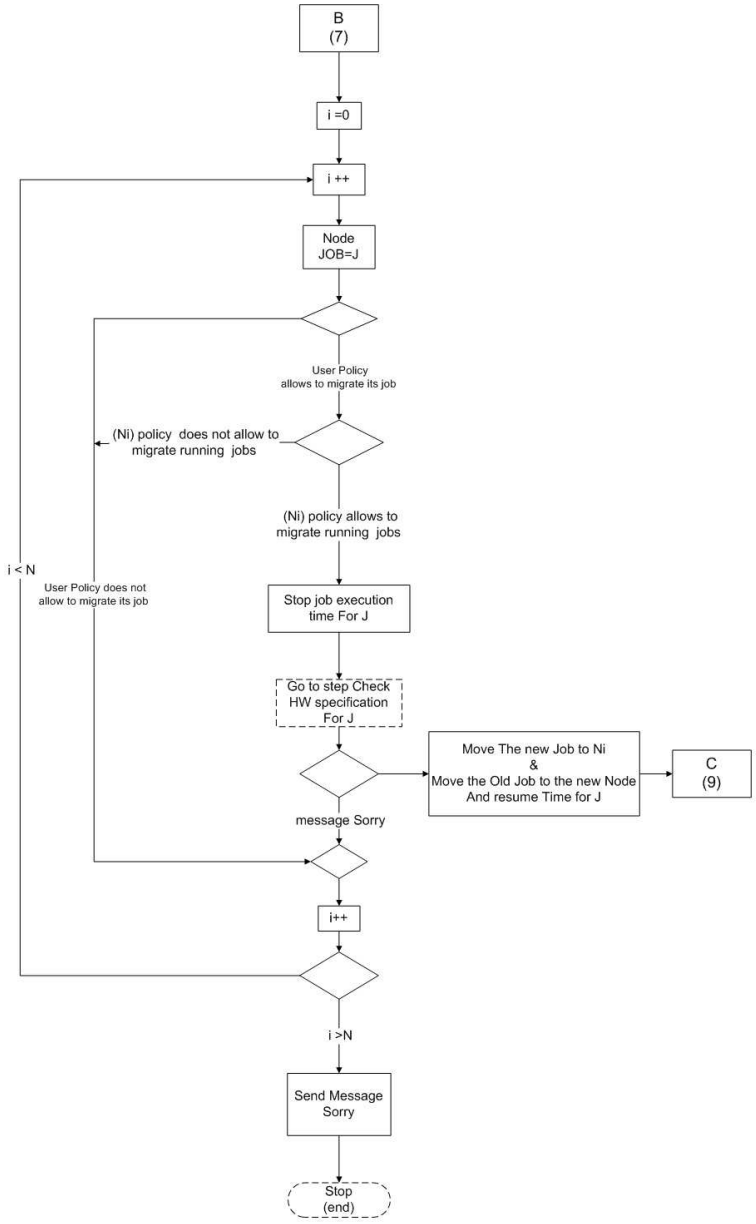

Figure 3.11: Job Migration Steps (7 and 8)

another, and the destination resource that fits the job hardware conditions and the availability time does not have the application software or data needed for the job(s), the resource broker will search the grid for the nodes that have this data/application software. This is can be done by requesting the information service and replica catalogue about the nodes that have these data or application

software's. Then it will ask the mobile policy agent to determine whether or not the data/application's software policy in these nodes allows their movements or copying from them. The Data/Application Software agent will check the policies for the nodes that contain the required data or application software and at the same time it will check the target resource to see if it allows moving or copying from the source node, and then return the results to the resource broker. If one of the nodes does support the mobility feature for data/application software, the resource broker will migrate or copy appropriately and send it to the resource that meets the job hardware and time requirements along with its policy. If all the nodes' policies do not support the data/application software mobility, the broker will tell the user that the grid cannot execute the job. Figure 3.9) shows the application software migiration, while Figures (3.12 and 3.13) shows the data migiration process (move or copy).

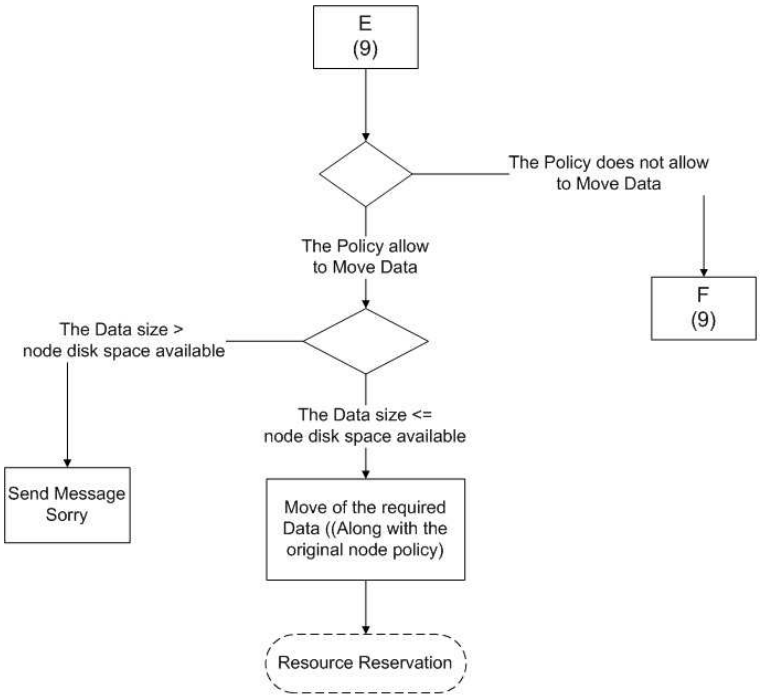

Figure 3.12: Data Migration (Move) Step (9)

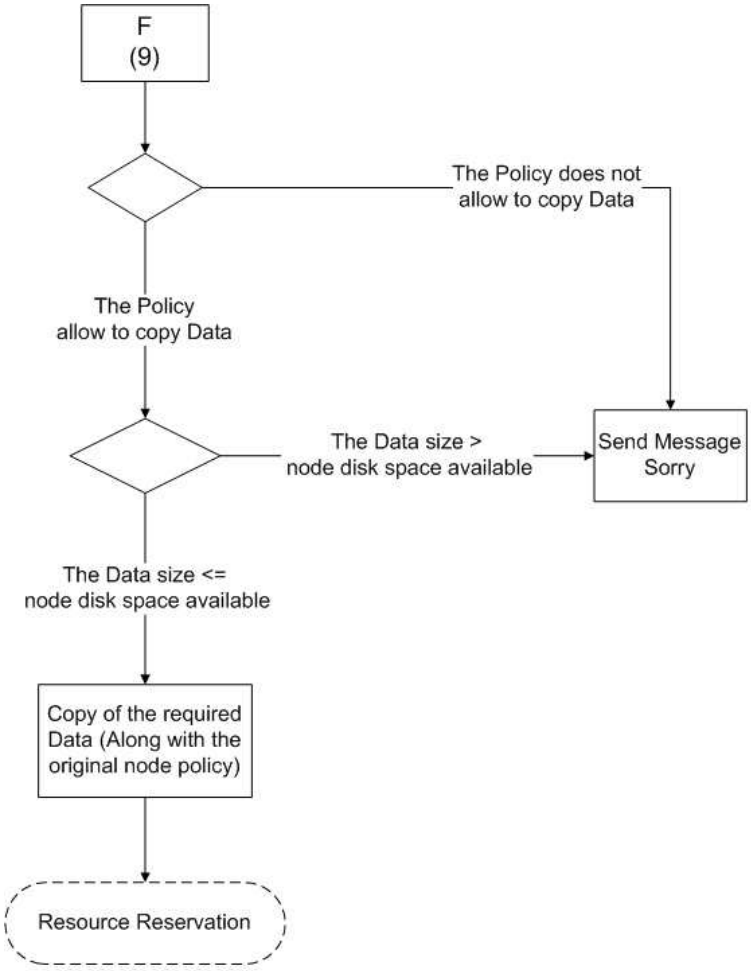

Figure 3.13: Data Migration (copy) Step (9)

3.6.3.4 Resource Reservation

Once resources that will accomplish the job's requirements have been found, the next step is to preserve these resources to run the job. If the application is parallel, the resource broker will co-allocate the needed resources to perform the jobs in parallel at the same application start time. But if the application is serial or network, the broker will preserve the needed resources with the needed time for every job determined by the JST for all jobs.

3.6.3.5 Monitoring and Job Live Time Organizing

Grid computing works in a dynamic environment in which resources are heterogeneous in nature and owned by various organizations and managerial domains. This problem has been dealt in our grid model by making every node in the grid able to transmit a periodic heartbeat to the resource broker. The heartbeat contains a timestamp, node name or ID and other optional information. If the resource broker does not sense a predictable heartbeat from a node, it makes that node in a SUSPECTED state and transmits it an Are-You-Alive message. If it replies, the resource broker makes it back in the ALIVE state and resumes as normal, otherwise the resource broker makes it in a FAILED state. This message assists the resource broker to determine the failure and monitoring grid resources.

An event is a message to point out that something has occurred at run time in a system. The event could be produced by resource broker, grid users, or grid nodes. Resource broker events are messages directed any modifications in job execution for the period of run time; they involve system performance deterioration, process and network failure and new nodes. User events allowing users to manage and control job execution for the duration of run time involve terminating job and reducing time. These messages are being transmitted from users to resource brokers. Resource events point out that something such as software and/or hardware failure has happened; therefore the resource has become unavailable. Also, these messages are sent from resources to resource broker.

Our resource broker deals with the event by user and/or grid policies. Users can describe events and how to handle the event, if they wish, at the job requirement description step. This can be done by identified message events, conditions and the event handler. This lets users manage job execution for the period of run time till terminating the job. They can also enforce conditions and handling events that respond to specific events with their related conditions. If users do not describe events, conditions and handlers, resource broker will use, by default, the grid policy.

3.7 Summary

This chapter described our new framework for policy management used for a single virtual organization and multiple virtual organizations; including design, forms and objectives. The former has been introduced in section two and there. The architecture presented in this chapter introduced the Grid Policy Agent, which enables the enforcement of mobility policies in the context of multiple VOs. This contribution is fundamental to the (how does the grid interact with policies for different domains and organizations in the case of mobile sharing and data movements?) and in particular the research aim one: Supports a multi-organization environment with different domains, combined with supporting the user preferences in its final decision and enforcing data policies in its designs (research aims three and four). This architecture has been published in Communications in the Computer and Information Science Conference (CoNeCo2011) [15] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110]. Section four and five gives an illustration that helps in highlighting the complicated operation a user would usually be subjected to when sending a job to the grid, and therefore the necessity for a grid resource broker to separate the user from such kinds of problems. Section six describes the architecture and functionality of the resource broker and the grid mobile policy agent that we built. How it locates appropriate resources for grid jobs using mobility feature to ensure the best achievement for these jobs and how the grid mobile policy agent plays a role in supporting the mobility feature. The framework that was introduced in the last section has been published in Risk and Security of Internet and Systems Conference (CRiSIS), IEEE Computer Society, 2011 [14] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

The most important concepts that have been described in this section are the job submission description language, as it will be extended to include policy statements in chapters 5 and 6, and the concept of mobility, as this work mainly considers policies associated with data jobs and resources that move inside and between Virtual organizations.

Chapter 4

A Computational Model

4.1 Introduction

This model is an outline of a computation executed on a specific architecture. As perceived by object-oriented paradigm the units in our computational model are objects that communicate and collaborate with each other. This model consists of objects and mechanisms. Section two presents these objects including node, data, policy, application software, grid job and grid application. While the mechanisms include timing, communication, termination, failure and mobility mechanisms is presented in section three. We will present a scenario to explain the advantage of mobility mechanism and the role of policy in it within grid systems in the same section.

4.2 Objects

This part will talk about the core of our computational grid. The objects of our computational model comprise node, data, policy, application software, grid job and grid application. The following, explains each one of them.

4.2.1 Node

A node is the most fundamental element in grid computing. It is a set of work units that can be shared and offer some facilities to the grid users. The main attribute of the grid nodes is the heterogeneity in capacity, speed, architecture and operating systems. Communication between nodes is accomplished using network resources like LAN and WAN. The main tasks for nodes are receiving, implementing and sending job's results. The user's requirements are different from one another; in order to execute the user's jobs every node may have its own application software and data to match these requirements. In our model we presume that all grid nodes have the ability to communicate with each other in grid environments to facilitate exchanging jobs, application software and data migration between nodes. Each node in the grid launches a regular heartbeat to the monitor which is in charge for managing and controlling grid nodes. This heartbeat includes a node name, job status (if the node is operating a job), timestamp and other optional data. Node can be either logical or physical node; a physical node may have one or more logical nodes. The logical nodes are designed to perform management roles for grid components and their relationships; such as Distributed File System (DFS) or distributed computer pool [54]. Physical nodes are classified according to their job in the grid. The two most familiar types of physical nodes are computation and storage nodes.

4.2.1.1 Computation Nodes

Computational nodes are devices that are utilized in relation to their hardware and processing abilities. The node could be a mainframe, cluster, high efficiency computer or a desktop. These abilities are primarily presented by the different types of processor models in which every computation node uses its own processor model and its own hardware designs and features such as speed, internal memory and software platform compatibility. Computation nodes can be utilized in many various techniques. A grid application can be implemented on one of the grid nodes rather than executing on a local node outside the grid. Parallel applications can be run on many processors; either they are situated on the same grid node or on many.

4.2.1.2 Storage Nodes

While the function of the computation node is handling jobs and applications, other devices have the ability for storing and offering data to other nodes inside the grid environment. These devices are called storage nodes. The most general storage type is secondary storage using hard disk drives or other storage media like tape drives. Applying secondary storage in a grid results in an enhancement in the performance, capacity, sharing and reliability of data swap within that grid. There are many types of file systems which deal with the storage and managing processes for the data among the nodes of the grid network. Such an example of these file systems are Distributed File System (DFS), Network File System (NFS) and General Parallel File System (GPFS).

4.2.1.3 Special Nodes

Grid computing has a range of implementations such as military, medicine and many different areas. These implementations asked for various architectures, equipment devices, capacities and special nodes to accomplish their jobs.

4.2.2 Data

Data is a part of information saved in a grid node, and is utilized by application software to accomplish specific jobs. The application software can then reach data whether on a local node or remotely. The data may previously be kept on one or more nodes, or it may arrive with the user job. In our model, we presume that every stored data owns its unique name to facilitate describing it by the users. One of the features of this data is the ability to mobile from one node to another according to the policy enforced by the resource broker. The owners of the data have the privilege to decide the policy for their data; this policy can be static or dynamic.

4.2.3 Policy

Policies are groups of rules and regulations created by one or more of jobs owners or administrators of nodes, describing how their jobs or resources can be dealt with and

used, how a work should be done, how security is employed in a domain and how an organization arranges and distributes their resources. Policies can be divided into user, grid and resource policies.

4.2.4 Application Software

Application software is one, or a group, of software programs that are located on grid nodes and can be used by grid users. This software is a set of instructions expressing a job, or group of jobs, to be performed by a node; it is moved into the node's RAM (Random Access Memory) and is carried out in the CPU (Central Processing Unit). The main task for application software is to execute and run the user job. It allows the end user to achieve one or more particular jobs by utilizing the resources of a computer in performing the conditions of a job that the user wants to achieve. Each one of these application software(s) has requirements and conditions; for example: CPU speed, disk space and operating system. In our model, application software should have the ability to accept the executable file (job), execute it and generate the result. Also it may have the ability to migrate between grid nodes according to the policy enforced by the resource broker in order to enable the needed node to achieve the user conditions for job execution. This feature helps in reducing the number of rejected jobs. The owners of the application software have the right to specify the policy for their application software. This policy can be static or dynamic. A static policy implies that the application software does not have the ability to migrate to other nodes in the grid. In a dynamic policy, application software has the ability to migrate there by moving or by copying itself.

4.2.5 Grid Job

A grid job is an individual element of work implemented by a suitable node in the grid. This job could be executing one or more system commands, operating machinery, moving or collecting data or simply calculating something. IBM [73] describes the job as "a single unit of work within a grid application. It is typically submitted for execution on the grid, has defined input and output data and execution requirements

in order to complete its task. A single job can launch one or many processes on a specified node. It can perform complex calculations on large amounts of data or might be relatively simple in nature." In our system, users need clearly to define their job requirements. These definitions may include job identification, resource specifications (CPU speed, physical memory), data staging, software applications and policy. This definition is essential, since it will determine where the job will be executed and how data is handled. The later is a very important issue; as it allows users to control their jobs during job execution. In our system, jobs can migrate from one node to another in the grid in any of the following situations:

- To clear the needed node from the existing job and make it available for another job.
- To accomplish the job if failure happens.
- To accomplish the job on a high performance node.

4.2.6 Grid Application

An application is a group of single jobs that work together to accomplish a mission. The expression "application" is used at the highest stage of work on the grid. IBM [73] describes the grid application as "a collection of work items to solve a certain problem or to achieve desired results using a grid infrastructure. For example, a grid application can be the simulation of business scenarios, like stock market development, that require a large amount of data as well as a high demand for computing resources in order to calculate and handle the large number of variables and their effects. For each set of parameters a complex calculation can be executed. The simulation of a large scale scenario then consists of a larger number of such steps. In other words, a grid application may consist of a number of jobs that together fulfill the whole task". There are five types of grid application:

- Grid applications that are used to solve very large problems (e.g.: the need for several CPUs and large memory). This type is called distributed supercomputing.

- Grid applications that are used in exploiting many suitable resources in order to improve aggregate throughput. This is a high throughput type.
- On demand grid Application. In this type remote resources are integrated for a particular amount of time.
- Data intensiveness. In this type grid applications are used to combine new information from large data sources.
- Grid applications which are used for Collaboration. Including supporting of communication between many participants.

Each one of these applications has requirements such as the names of all integrated jobs, the application flow (relations between these jobs) and the requirements of each single job. These application requirements are very important issues for the resource broker; as it helps in selecting the suitable resources that can meet applications requirements.

Application flow denote the links between jobs that meet the general requirements of the application. The three types of application flow are parallel, serial and network applications. Parallel applications include several jobs that can be implemented simultaneously. One of the advantages of such applications for grid computing is increasing scalability. Serial application flow is a single sequence of job execution where every job has to wait for its predecessor to be completed and deliver output data as input to the next job. Network application includes a number of jobs, some of which are executed in parallel whilst others are interdependent. In other words, an application that includes combined parallel and serial applications.

4.3 Mechanisms

4.3.1 Timing Mechanism

Time is crucial issue in grid computing, its aim is to utilize underutilized resources to reach faster job execution times. Every node in our structural design has its own internal clock. Each node must then regularly synchronies its clock with that of resource

broker by using network time protocol. Also they have to state their availability time when they participate in the grid. The main tasks of the timing mode is to maintain and control system time, and that would prevent jobs from operating longer than they are permitted to. In addition, it assists in the case of handling failure. To implement a job, a certain period of time is needed; the majority of resources apply time as a charging unit because it is easily quantifiable. For that reason it is easy for jobs to have Worse Case (WorseCase).

Worse Case is the maximum time that the job can be on hold until it is executed. Several issues such as heterogeneous environments, communication delays and data transfer speeds cause difficulties for the determination of job execution times.

4.3.2 Communication Mechanism

Communication is responsible for linking objects in the grid environment with each other to perform their actions with the aim of flexibility. This communication is for the purpose of swapping and the transmitting of information and data between these objects. Moreover, Information transmission is essential when dependencies exist. Also communication assists in determining failures. Communication between objects is accomplished using Remote Procedure Calls (RPC). This communication can be either synchronous or asynchronous. Our model uses asynchronous communication, which leads to a non-blocking send.

Asynchronous communication is a substitute form that may be helpful in cases when an allowance is made for an object to recover replies later. Communication presents the method of organization and management between objects. Because our model is client/server architecture, the client sends requests to a grid service by using a remote procedure call. When the request has been completed, notification is returned to the client, who can at the same time generate a new remote procedure call to the similar service.

4.3.3 Termination Mechanism

Once a job has been submitted, it starts processing on the nodes till finish. Usually a job finishes as a result of circumstances such as normal, user termination or failure. Our model uses a termination that reserves time and lets objects operating in parallel to accomplish one task; when any object has completed its job, it does not wait or go on hold for others to complete their jobs. This type of termination is called conventional termination (being opposed to distributed termination).

4.3.4 Failure Mechanism

Failures in distributed systems can be irregular in that they can dump the job in one of many failed cases. In the grid system the possibility of failure is strong, for the following reasons:

- Grid environments are heterogeneous. That includes entities, hardware and software components. All of that can lead to failure when they interact.
- Grid environments are very dynamic, with objects regularly joining and leaving the system.
- The difficulty in detecting failures in a dynamic and heterogeneous system.

The failure can happen because of many reasons such as, software or hardware failure, communication delays, process failure and network failures. Failures in grid computing are partial, that means particular components fail while others still function. When hardware or software faults happen, jobs may generate wrong results or hold before they are finished. Our model presumes that all system components, either hardware or software, may fail at any time.

Fault tolerance is a vital attribute and an essential function for grid environments so as to prevent losing the computation time. Two process support a fault tolerance mechanism: failure detection and failure handling. Detection is connected to the heartbeat that is sent from the nodes to the resource broker at a regular basis. Each heartbeat includes a timestamp, job status (if the node still processing a job), node name and other

optional information. If resource broker does not sense an expected heartbeat from any node, it places that node in a SUSPECTED state, and transmits an are-you-alive message to the node. If the node replies with a message, in that case the resource broker puts the node back in the ALIVE state and carries on as normal. Otherwise, the broker puts the node in a FAILED state. This message assists the broker to detect the failure and recover it. When the node transmits a heartbeat to the resource broker, the latter will match the existing state of the job with the most recent state; if it is the same, the broker will presume that the job has failed.

If failure happens, the resource brokers will response by moving or migrating the job with its current status to another appropriate resource on the list as found by the resource broker at the resource discovery level, to restart processing the job from the point of failure. If only the failed resources exist in that list, or if all the listed resources are presently not free, the user will be informed by the resource broker that the grid is unable to execute your job because of failure. Check-pointing schemes let a job resume from the point of failure, avoiding the need of rerunning the whole job. By returning to a previous checkpoint, a system can reload the earlier state and continue computation from the point of failure. In our architecture, the system can detect and handle, and thereby recover from the failure state.

4.3.5 Mobility Mechanisms

Mobility is the ability of physical or logical (virtual) computational resources (software code, running object, portable notebook PC's, data and mobile agent) to migrate from one site to another through local or global networks. David G. Rosado and others in [101] described the mobility as "In the purview of Grid and Mobile Computing, Mobile Grid is an heir of the Grid, which addresses mobility issues, with the added elements of supporting mobile users and resources in a seamless, transparent, secure and efficient way [[63]; [74]; [93]]".

In our module we have utilized weak and strong mobility. Weak mobility allows code to move through the networks. In some cases the codes have initial data assigned but without execution states (for example the state of the computation is lost at the

first node). Strong mobility is the ability of a computational environment to mobile the code and execution state (the context of execution) to start again at a new resource. The execution state comprises of running code, saved processor registers, program counter, local variables and return addresses. A set of organizing execution controls operates in a process on the user machine, and then accesses the remote resource by the invocation of a remote process. One of the most important features of strong mobility is that jobs can choose to migrate between sites while it is processing information, most likely decreasing the distance between it and the next group of data it intends to process. The reliability challenge of partial failure is decreased because the job state is only in one site at a time.

To explain the advantage of mobility mechanism and the role of policy in it within grid systems, we will consider the following scenario, which is divided into three sections.

- **First section: Grid Resources Specifications**

  The grid contains five nodes; each node has a different specification. These specifications are: hardware, domain, application software, data and policies. Also it contains the running jobs, if presented, as shown in Tables (4.1), (4.2) and (4.3).

- **The second section: Jobs Requirements**

  Three jobs need to be executed by the grid resources. The requirements needed to accomplish each job include hardware, software, input, output, dmoanin and policies, as shown in Tables (4.4) and (4.5).

- **The third section: Fits the Jobs Requirement to Grid Resources**

  As mentioned before, the resource broker is responsible for locating the optimal resource that can meet the job requirements and scheduling the jobs into grid resources. This is complex issue since job requirements are various and complicated and there is a shortness of resources, especially for data and application software, in addition to the diversity of resource policies. All these reasons will raise the number of rejected jobs.

  Mobility is an effective approach to resolve this problem, because it will develop

the node in order to have the ability to fit the job requirements. This is can be done by moving (migrating) the needed data or application software to the required node to be able to accomplish the job. Or migrating the job itself from one node to another empty one, with respect to the policies, to make the required node available for another job. All of these issues are illustrated in the following Figures (4.1) and (4.2).

Table 4.1: Grid Nodes Hardware Specification

Node Name	Hardware				Domain
	CPU		Memory		
	Speed(GHz)	Count	RAM(Mega)	Shared or Disturbuted	
Node 1	1	1	1024	-	USA
Node 2	1	1	2048	-	Japan
Node 3	2	2	2048	D	UK
Node 4	1	1	2048	-	China
Node 5	1	1	2048	S	UK

Table 4.2: Grid Nodes Application/Data Specification

Node Name	Application				Data	
			Requirement			
	File	Version	CPU Speed(GHz)	Disk Space(Mega)	FileName	Size(Mega)
Node 1	S1	9.2	0.5	200	D2	1000
	S2	1.0	0.5	700		
Node 2	S5	2	1.0	300	D1/D2	900
Node 3	S4	1.0	1.0	500	D3	700
Node 4	S3	1.0	1.0	900	-	D1
	S4	5.0	1.0	250		
Node 5	S3	1.0	1.0	500	D1	700

- **Job Migration**

  From Tables (4.1, 4.2, 4.3, 4.4 and 4.5) it can be seen that Job1 requirements fit the Node1 specification, but Node1's policy is to allow only single job to run at any time (exclusive execution), so there is a need to migrate the existing job (Job 2) on Node1 to another node that fits Job2 requirements. The resource broker looks for this substitute node and finds Node4 and Node5; but Node4 domain is

Table 4.3: Grid Nodes Policy Specifications and Running Jobs

Node Name	Policy					Jobs Running
	Exclusive Execution	Move Data	Move Application	Move Job	Restricted (Domain/User/Job)	
N1	Yes	Yes	Yes	Yes	China/U4	Job2
N2	No	No	Yes	Yes	China	-
N3	Yes	Yes	No	Yes	None	-
N4	Yes	Yes	Yes	Yes	None	-
N5	Yes	Yes	Yes	Yes	None	-

Table 4.4: Job Specification and Application Software Requirements

User Name	Job Name	Node Specification				Application Software		Data
		CPU		Memory		Name	Version	
		Speed(GHz)	Count	RAM(Mega)	S/D			
U1	Job 1	1	1	1024	-	S2	1	-
U2	Job 2	1	1	2048	-	S3	1	D1
U3	Job 3	2	2	2048	-	S4	1	D2
U4	Job 4	1	1	2048	-	S4	5	D2
U5	Job 5	2	2	2048	-	S5	1.1	D3

Table 4.5: Job Domain/Policy

User Name	Job Name	Policy			Domain
		Exclusive Execution	Move Job	Restricted (Domain/User)	
U1	Job 1	No	Yes	None	China
U2	Job 2	No	Yes	China	UK
U3	Job 3	No	No	None	USA
U4	Job 4	No	No	None	USA
U5	Job 5	No	No	None	UK

in China which is against the policy of Job2 and Node1 policy. Therefore, the resource broker sends job1 to Node1 that says "send Job2 together with its status (memory image) to Node5 for execution".

- **Data Migration (case 1)**

  As shown in Tables (4.1, 4.2, 4.3, 4.4 and 4.5), Job3's requirements fit Node3's specifications, but Node3 does not contain data (D2); this data is available in Node1 and Node2, Node1 policy is to allow movement of this data as well as Node3's data requirements, while Node2 is not. The resource broker will there-

fore send a message to Node3 telling it to take data (D2) along with its policy from Node1 and execute Job3.

- **Data Migration (case 2)**

  As shown in Tables (4.1, 4.2, 4.3, 4.4 and 4.5), Job4's requirements fit Node4's specifications, but Node4 does not contain data (D2); this data is available in Node2 and Node3(after migration). Node2 policy is not to allow movement of data to china domain, but the policy in Node3 allows this kind of movements, but the data in Node3 was moved originally from Node1 which its policy does not allow to move data to China domain. Therefore, the resource broker will send a message to User4 saying that the grid is unable to execute Job4, because the needed data is unavailable.

- **Application Software Migration**

  Tables (4.1, 4.2, 4.3, 4.4 and 4.5), show that Job5's requirements fit Node3's specifications. But Node3 does not have application software (S5). Node2 does, however, and its policy is to allow this application software as well as node3's application software requirements. The resource broker will therefore send Job5 with a message to Node3 telling it to take application software (S5) from Node2 and execute Job5.

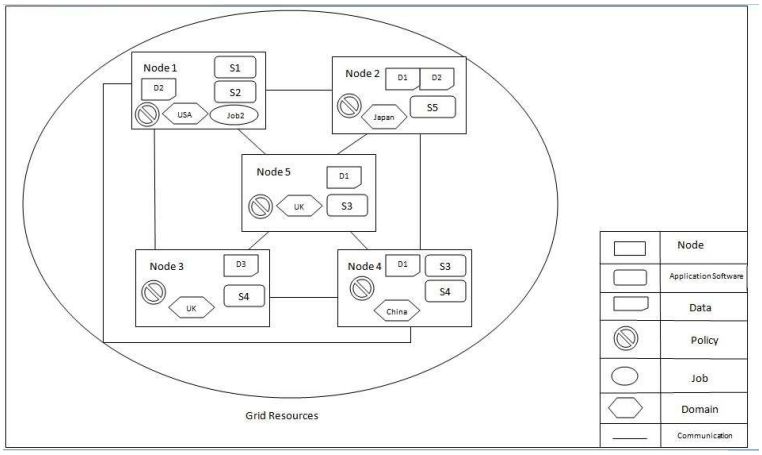

Figure 4.1: Grid Resources (Infrastructure)

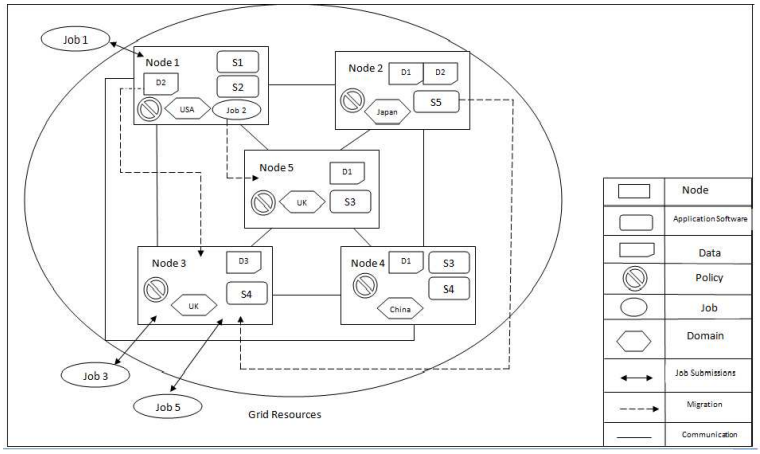

Figure 4.2: Grid Resources after Mobility

4.4 Summary

This chapter explained the activities of the system which composes of computational model objects and mechanisms. Objects are nodes, data, policy, application software, grid jobs and grid applications as illustrated in section two. Whereas mechanism manages and controls system elements to accomplish all the aims of our grid structure, it includes: timing mechanism, which is to utilize underutilized resources to reach faster job execution times. Communication mechanism which is responsible for linkingobjects in the grid environment with each other to perform their actions with the aim of flexibility. Termination mechanism which is the way to terminate the job after fulfils its requirements. Fault tolerance mechanism which is a vital attribute and an essential function for grid environments so as to prevent losing the computation time. And finally, mobility mechanism mainly considers policies associated with data jobs and resources that move inside and between Virtual organizations. All of former mechanisms are illustrated in section three.

Chapter 5

Grid And Job Description Language

5.1 Introduction

Grid computing supports and coordinates the sharing of heterogeneous resources that are distributed across multi-organizations. Many languages have been presented for this propose such as Globus European Data Grid JDL [92] and Resource Specification Language (RSL) [2]. The Job Submission Description Language (JSDL) [11] is considered one of the modern languages that is created for grid description issue.

Our system converts user preferences and conditions requirements to an External-JSDL that expresses users' jobs requirements and is understood by the grid environment no matter what domain the resources are. Also the proposed system converts the External-JSDL requirements to an Internal-JSDL; as a language that is used to communicate between the resource broker and the grid nodes, and between the grid nodes themselves in different organizations and domains. The system stores these language expressions as an XML schema in order to be retrieved later and sends the expressions to Jade to be simulated. The reason behind using XML as a language for grid and job requirement expression is that XML has many attractive attributes such as simplicity in reading,

understanding and processing by users and computers. The External-JSDL language is introduced in this chapter, whilst the Internal-JSDL language has been introduced in the following chapter.

This chapter is organized as follow: the second section gives an overview of our language which is considered one of the contributions in this thesis. The third section presents the structure of this language. Section four presents the External-JSDL language which is considered a language to express the grid users' preferences to the grid, followed by a section that illustrates its structure.

5.2 Language overview

Our grid system has been simulated by using Jade simulator, which is a software framework fully implemented in Java language that allows agents to execute tasks defined according to pre-defined policies. We designed interfaces that help grid administrators build their grid with all of it resources. At the same time these interfaces give the grids' users the ability to describe and send their jobs to the grid. When the grid administrators or grids' users submit their requirements by using the previous interfaces in order to be simulated by Jade, our system converts these requirements to an XML language and then sends them to Jade. At the same time, the system stores these XML files to help in retrieving the created grid later. Our system converts the user preferences and conditions requirements to an External-JSDL that expresses the users' jobs requirements and can be understood by the grid environment no matter what domain the resources are. Also the proposed system converts the External-JSDL requirements to an Internal-JSDL; a language that is used to communicate between the resource broker and the grid nodes, and between the grid nodes themselves in different organizations and domains.

5.3 Structure of the language

5.3.1 Grid Structure Language

When creating a grid, we need to define its name, nodes, the polices that control their action and define the relations between its components. Figure (5.1) shows these components.

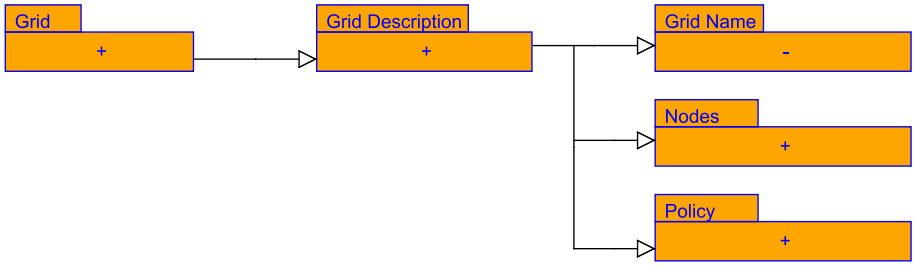

Figure 5.1: A Grid Structure Schema

The grid elements are written as XML schema applying the BNF-style standard for elements and attributes. According to this style the character '?' symbolizes the existence of zero or one. The character '+' symbolizes the existence of one or more and '*' is a symbol referring to existence of zero or more. There are many normative XML schema modes such as normalisedString, string, any##other .. etc. The following is the description of the pseudo schema:

```
<Grid>
        <GridDescription>
                <Name.../>?
                <Nodes.../>+
                <Policy.../>?
        </GridDescription>
        <xsd:any##other/>?
</Grid>
```

5.3.1.1 Grid Name

The name of the grid has to be unique, and this component is a string one with a multi-divergence of zero or one.

5.3.1.2 Nodes

After determining the grid name, the grid administrator defines the grid nodes. These nodes are responsible for hosting the users' jobs and fulfilling their requirements. Each node may have at least single application software and/or data (or more). The regulations that control the relations between the node resources (hardware, application software and data) and user's jobs are created by the grid administrator.

5.3.2 Policy

A compound component with a multiplicity of zero or one; it is used to describe the grid's policies. It supports the following elements, shown in Figure (5.2):

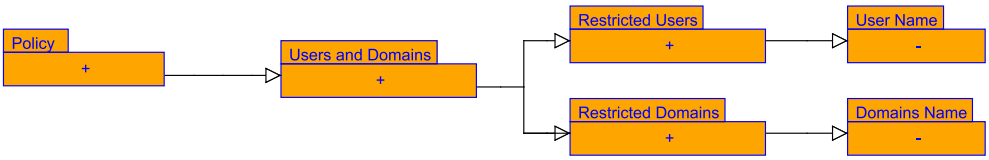

Figure 5.2: Grid Policy Schema

- **Pseudo Schema**

```
<Policy>
        <UsersandDomains/>
                <RestrictedUsers>?
                <RestrictedDomains/>?
        </UsersandDomains/>?
</Policy>?
```

1. **Users and Domains** refers to a compound component with a multiplicity of zero or more. It states the name of the users and/or domains that are not allowed to execute or mobile their jobs in the grid. It should have the ability to support the following components.

   (a) **Restricted Users** represents a compound component with a multiplicity of zero or more. This states the name of the users that are not allowed to execute or mobile their job in the grid. It has to support the User Name component.

   - **User Name** is a string with a multi-divergence of one or more. It states the User names. It can be either a node name or logical set of users (cluster).
   - **Pseudo Schema**

```
<RestrictedUsers>
        <UserName></UserName>
</RestrictedUsers>?
```

   (b) **Restricted Domains** refer to a compound component with a multiplicity of zero or one. This states the name of the domains that are not allowed to execute or mobile their jobs in the grid. It has to support the Domain Name component.

   - **Domain Name** is a string with a multi-divergence of zero or more. It states the domains' names. It can be either a domain name or logical set of domains (cluster).
   - **Pseudo Schema**

```
<RestrictedDomains>
        <DomainName></DomainName>
</RestrictedDomains>?
```

5.3.3 Node Structure Language

In this section we have configured our Java interfaces so that they can be easily handled by the grid administrators in order to create their grid nodes. The creation is divided into four parts; the first one deals with the node hardware specification. The second part deals with the application software(s) that is available on that node. The third part deals with data that are owned by that node and the final one deals with the policies and regulations for the previous elements. As soon as the grid administrators submit their node requirements to the Jade through the Java interfaces, these requirements are stored as XML schema to be used by our system to retrieve the created grid later. Figure (5.3) shows node components followed by an explanation of the language used for each one of them.

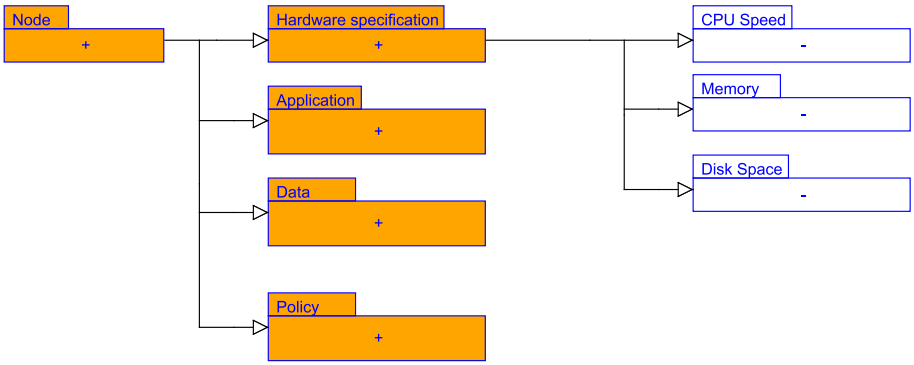

Figure 5.3: Node Structure Schema

- **Pseudo Schema**

```
<Node>
        <Hardwarespecification/>?
        <Application/>?
        <Date/>?
        <Policy/>?
</Node>+
```

5.3.3.1 Node Hardware Specification

Hardware specification elements are a compound element with a multi-divergence of zero or one; it is used to define the hardware available in the node. Figure (5.3) shows these elements.

- **CPU Speed** is a variety value with a multi-divergence of zero or one that determines the speed of CPU available in the node. It is demonstrated in Hertz.
- **Physical Memory** is a variety values with a multi-divergence of zero or one that states the amount of physical memory available in the node. Physical Memory is given in bytes.
- **Disk Space** is a variety value with a multi-divergence of zero or one that specifies the amount of disk space available in the node. It is given in bytes.
- **Pseudo Schema**

```
<HardwareSpecification>
        <CPUSpeed/>?
        <Memory/>?
        <DiskSpace/>?
</HardwareSpecification>?
```

5.3.3.2 Node Application

A string component with a multi-divergence of zero or more; this is used to determine the application's name available in the node along with its policy as shown in Figure (5.4). In the policy part the administrator can choose if the application software can be moved or copied within the grid environment in respect to the restricted users and domains field.

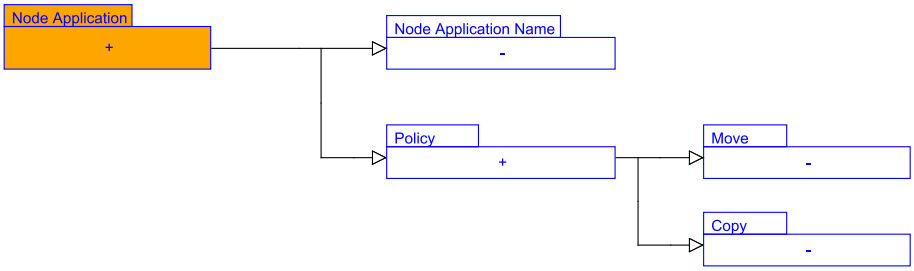

Figure 5.4: Node Application Software Schema

- **Pseudo Schema**

```
<NodeApplication>
        <NodeApplicationName.../>
                <ApplicationName.../>*
        </NodeApplicationName.../>?
        <Policy>
                <Move.../>?
                <Copy.../>?
        </Policy>?
</NodeApplication>?
```

5.3.3.3 Node Data

This section is related to data that should be available in the node that is going to execute the job. It contains two elements called Data Name which is a string component with a multi-divergence of zero or more that describes the local name of the file or directory on the node that is holding data as shown in Figure (5.5). In the policy part the administrator can choose if the data has the ability to be moved or copied within the grid environment in respect to the restricted users and domains

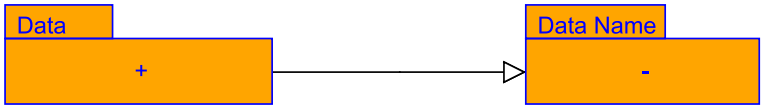

Figure 5.5: Node Data Schema

- **Pseudo Schema**

```
<NodeData>
        <DataName.../>
                <Name.../>*
        </DataName.../>?
        <Policy>
                <Move.../>?
                <Copy.../>?
        </Policy>?
</NodeData>?
```

5.3.3.4 Policy

It is a compound component with a multiplicity of zero or one; it is used to describe the node's policies. It supports the following elements, as shown in Figure (5.6):

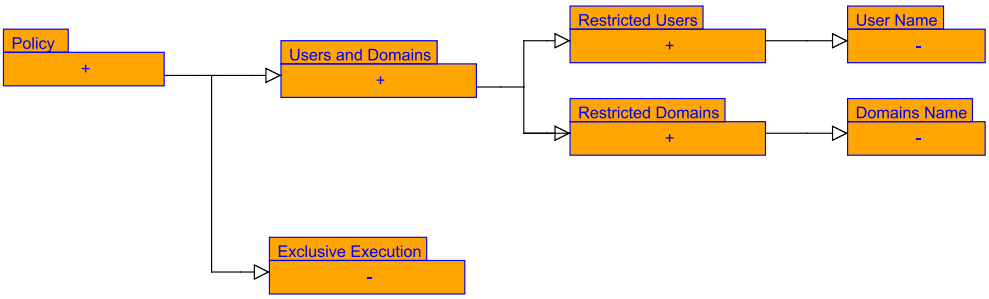

Figure 5.6: Node Policy Schema

- **Pseudo Schema**

```
<Policy>
        <UsersandDomains/>
                <RestrictedUsers>?
                <RestrictedDomains/>?
        </UsersandDomains/>?
        <ExclusiveExecution/>?
</Policy>?
```

1. **Users and Domains** is a compound component with a multiplicity of zero or one. This states the name of the users and/or domains that are not allowed to execute or mobile their jobs in the node. It should have the ability to support the following components.

   (a) **Restricted Users** is a compound component with a multiplicity of zero or one. It states the name of the users that are not allowed to execute or mobile their job in the node. It has to support the User Name component.

   - **User Name** is a string with a multi-divergence of zero or more. It states the User names. It can be either a node name or logical set of users (cluster).

   - **Pseudo Schema**

```
<RestrictedUsers>
        <UserName></UserName>*
</RestrictedUsers>?
```

   (b) **Restricted Domains** is a compound component with a multiplicity of zero or one. It states the name of the domains that are not allowed to execute or mobile their jobs in the node. It has to support the Domain Name component.

- **Domain Name** is a string with a multi-divergence of zero or more. It states the domains' names. It can be either a domain name or logical set of domains (cluster).
- **Pseudo Schema**

```
<RestrictedDomains>
        <DomainName></DomainName>*
</RestrictedDomains>?
```

2. **Exclusive Execution** is a Boolean with a multi-divergence of zero or one that determines if the job allows only itself to run at any times on the allocated resources or not. In the true option, the job should be executed on the chosen resource exclusively.
   - **Pseudo Schema**

```
<ExclusiveExecution>xsd:boolean</ExclusiveExecution>?
```

5.4 External-JSDL Overview

Before submitting a job to the grid resources for a computational process, a job needs to be described by a language that can be understood by different machines and domains. External-JSDL expresses the requirements of computational jobs particularly in grid environments. It is based on XML description language created by the Open Grid Forum (OGF). External-JSDL does not only describe the requirements of computational jobs, but can also describe requirements that are not essential to grid computing. In other words, External-JSDL is not limited to grid computing.

The design of this language makes it easy to be merged with other languages to solve complex jobs. The motivations behind creating External-JSDL as one of the well

known description languages in grid environment are the fact that there are many different organizations which have different job management systems. Each system uses its own description language for describing job requirements. This makes it difficult to co-ordinate between these multi organizations systems. Also for purpose of finding the suitable resources to accomplish the grid user's job, the job description may have to pass between different systems to find the best resource requirements for the job. External-JSDL is created as a proposed solution to solve these problems because of its features of mapping between different systems.

5.5 External-JSDL Structure

There are five major External-JSDL components shown in the architecture in Figure (5.7). The External-JSDL XML schema comprises of the core element (JobDefinition), which includes only one compulsory child element labeled JobDescription. The latter includes: JobIdentification which is used to distinguishe the job from others; Application which is used to define the application software; Hardware Specifications to define the hardware requirements for the job to be executed on a specific resource(s); Data which is used to describe the data that should be existed in the execution host to accomplish the job execution; Policy to decide how security is applied in a domain and defines the user job limitation and regulations conditions and Job Data for describing the data that is being sent along with the job description from the user side.

The External-JSDL elements are also written in XML schema and apply BNF-style standard for elements and attributes. The following is the description of the pseudo schema:

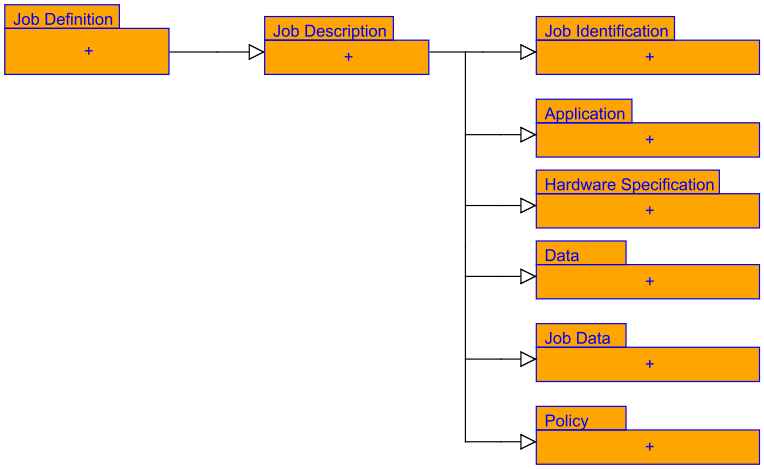

Figure 5.7: External-JSDL Structure Schema

```
<JobDefinition>
        <JobDescription>
                <JobIdentification.../>?
                <Application.../>?
                <HardwareSpecification.../>?
                <Data.../>?
                <Policy.../>?
                <JobData.../>?
        </JobDescription>
        <xsd:any##other/>?
</JobDefinition>
```

5.5.1 Job Identification

This is a string component used by the user to define the job name. It has not to be unique, and has a multi-divergence of zero or one. It includes Job Name as shown in Figure (5.8).

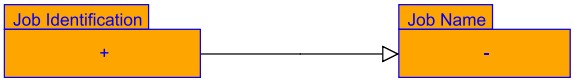

Figure 5.8: Job Identification Schema

- **Pseudo Schema**

```
<JobIdentification>
        <JobName></JobName>?
</JobIdentification>
```

5.5.2 Application

To execute a job on a specific resource, this resource should have the appropriate software. If the user is not specified, the External-JSDL document identifies it as a null job. To identify a software application, it is essential to specify the application's software name.

Application Name is a string component with a multi-divergence of zero or more, used to determine the application's name as shown in Figure (5.9).

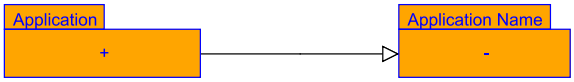

Figure 5.9: Application Software Schema

- **Pseudo Schema**

```
<Application>
        <ApplicationName.../>*
</Application>?
```

5.5.3 Hardware Specification

To achieve successful job execution, grid resources have to be "well" described by the grid users. Hardware specification elements are a compound element with a multi-divergence of zero or one; it is used to define the hardware requirements for the job. Figure (5.10) shows these elements followed by a short description for each one.

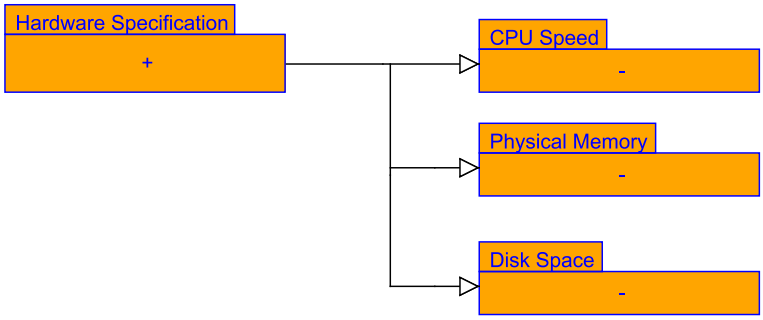

Figure 5.10: Resource Schema

- **CPU Speed** is a variety value with a multi-divergence of zero or one that determines the speed of CPU needed for the job. It is given in Hertz.
- **Physical Memory** is a variety values with a multi-divergence of zero or one that states the amount of physical memory needed by the job. Physical Memory is given in bytes.
- **Disk Space** is a variety values with a multi-divergence of zero or one that specifies the amount of disk space needed by the job. It is given in bytes.
- **Pseudo Schema**

```
<HardwareSpecification>
        <CPUSpeed/>?
        <Memory/>?
        <DiskSpace/>?
</HardwareSpecification>?
```

5.5.4 Data

For each job submitted to the grid for execution, data may be required or should be available in the host that is going to execute the job. This data should be described carefully by the user prior to the job's execution starts. It contains one element called Data Name which is a string component with a multi-divergence of zero or more. This element describes the local name of the file or directory on the host that is holding the execution as shown in Figure (5.11):

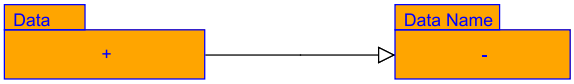

Figure 5.11: Data Schema

- **Pseudo Schema**

<Data>

<DataName.../>*

</Data>?

5.5.5 Job Data

This component describes the data that have been sent from the user side to the grid along with the job description elements. These data are needed to accomplish the job. This component contains one element (Data Name) which is a string component with a multi-divergence of zero or more. Figure (5.12) shows this element.

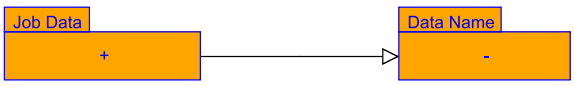

Figure 5.12: Job Data Schema

- **Pseudo Schema**

```
<JobData>
                <JobDataName.../>*
</JobData>?
```

5.5.6 Policy

Policies are groups of regulations, standards and practices written by grid users about how their jobs can be handled and used. They decide how a job should be done, how security is applied in a domain and how an organization organizes, secures and distributes its resources.

Policy is a compound component with a multiplicity of zero or one; it is used to describe the user's policy for the job. It supports the following elements, as shown in Figure (5.13):

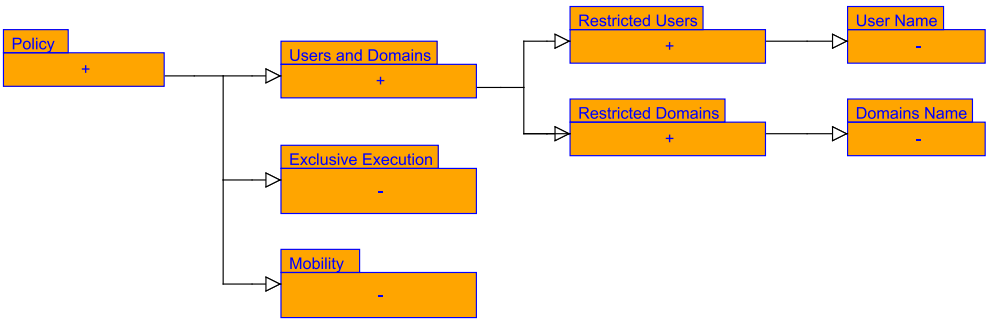

Figure 5.13: Policy Schema

- **Pseudo Schema**

```
<Policy>
        <UsersandDomains/>?
        <ExclusiveExecution>?
        <Mobility/>?
</Policy>?
```

1. **Users and Domains** refer to a compound component with a multiplicity of zero or one. It states the name of the users and/or domains that are not allowed to execute or mobile the user's job in it. It should have the ability to support the following components.

   (a) **Restricted Users** is a compound component with a multiplicity of zero or one. It states the name of the users that are not allowed to execute or mobile the user's job in it. It has to support the User Name component.

   - **User Name** is a string with a multi-divergence of zero or more. It states the User names. It can be either a node name or logical set of users (cluster).
   - **Pseudo Schema**

```
<RestrictedUsers>
        <UserName></UserName>*
</RestrictedUsers>?
```

   (b) **Restricted Domains** is a compound component with a multiplicity of zero or one. It states the name of the domains that are not allowed to execute or mobile the user's job in it. It has to support the Domain Name component.

   - **Domain Name** is a string with a multi-divergence of zero or more. It states the domains' names. It can be either a domain name or logical set

of domains (cluster).

- **Pseudo Schema**

```
<RestrictedDomains>
        <DomainName></DomainName>*
</RestrictedDomains>?
```

2. **Exclusive Execution** is a Boolean with a multi-divergence of zero or one that determines if the job allows only itself to run at any time on the allocated resources or not. In the true option, the job should be executed on the chosen resource exclusively.

   - **Pseudo Schema**

```
<ExclusiveExecution>xsd:boolean</ExclusiveExecution>?
```

3. **Mobility** is a Boolean with a multi-divergence of zero or one that determines if the job and the job data allow moving (mobile) between hosts. In the true choice, the job and job data should have the ability to mobile between hosts according to the User and Domain Name components.

   - **Pseudo Schema**

```
<Mobility>xsd:boolean</Mobility>?
```

5.6 Summary

This chapter presents an overview of how grid administrators and users have the flexibility to configure their grid environment. Through this chapter we introduced our language to configure the grid system using the XML language as in section two and three. This language is considered one of our contributions in this research. Also we

introduced the External-JSDL, which is used to define users grid jobs and preferences by presenting their identifications, application software, resource specifications, data, user data and policies that are needed to accomplish the user's job, as presented in section four.

The reader needs to bear in mind the following terms: External-Job Submission Description Language (JSDL) and Job Mobility as a language that expresses the users' jobs requirements and can be understood by the grid environment no matter what domain the resources are. In the next chapter we will propose the Internal-JSDL; as a language that is used to communicate between the resource broker and the grid nodes, and between the grid nodes themselves in different organizations and domain. Then the system stores these language expressions as an XML schema in order to be retrieved later and sent to Jade to be simulated. The reason behind using XML as a language for grid and job requirements expression is that XML has many attractive attributes such as the simplicity in reading, understanding and processing by users and computers.

Chapter 6

Mobility And Grid Components Language

6.1 Introduction

In the previous chapter (Chapter 5) we have discussed the language that is used to create and configure the grid environment and the External-JSDL (job description) as a language that enables the grid users to express their jobs and preferences to the resource broker. But there is a need for a language to support the communications between the resource brokers and grid resources, and between the resources themselves. We have presented in this chapter a new language that can support this communication; it is called Internal-JSDL. It is based on an XML language which defines how a job must operate on the resources. It defines the conditions and requirements for this job such as data, application software, mobility and the policies related to each one of them. Internal-JSDL language is different from other languages in that it supports the mobility feature including Job, data and application software. The resource broker receives External-JSDL specifications described by the grid user and translates them into a clear specifications language used to find available resources to fit those requirements. These specifications are coded in Internal-JSDL as shown in Figure (6.1).

The Internal-JSDL is subjecte to the External-JSDL. Also it is subject to resource/grid policies and resource availability. Internal-JSDL components crack into the following categories:

- Job identification specifications
- Data specifications
- Mobility modes and conditions
- Policy
- Software specifications
- Output specifications

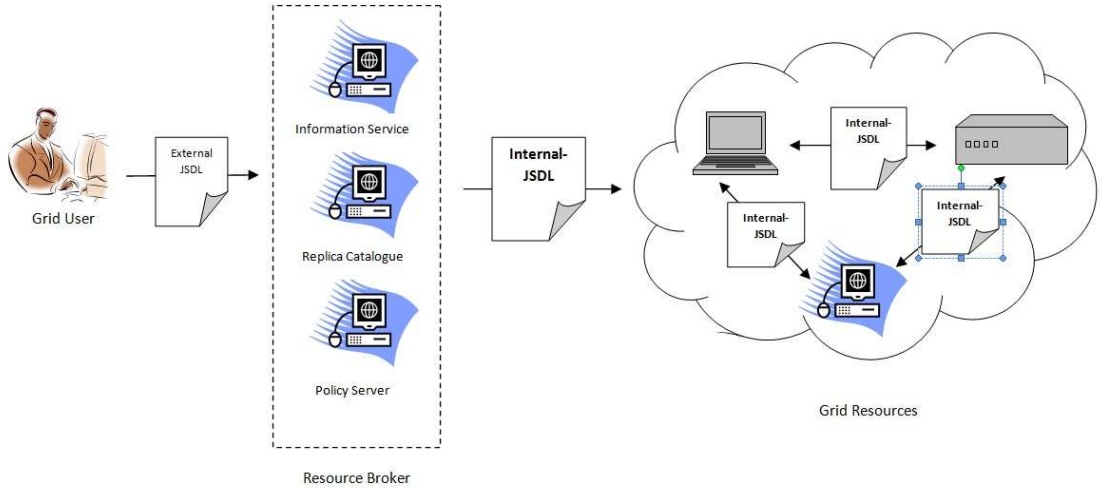

Figure 6.1: External-JSDL and Internal-JSDL

This chapter is considered one of our contributions in this thesis and it is organized as follows; Section two presents an overview for our language that can enable communication between the scheduler (resource broker) and grid nodes and between the nodes themselves. In the next sections we will discuss the language that is used to express the (jobs, data, mobility, policy and application software) in this new language. Each one of them is discussed in separate section. And finally the last section introduces the

output which is responsible for determining the files that should be transferred (stag out) from the host that is holding the job execution.

6.2 Internal-JSDL Structure Components

6.2.1 Structure

The Internal-JSDL main components are organized as follows: The main component MJob includes one or more compulsory child components named job. The job component has attributes that state the name of the node (grid resource), JobId, Data, Mobility, Policy, ApplicationSoftware and Output. Figure (6.2) shows the schema of Internal-JSDL.

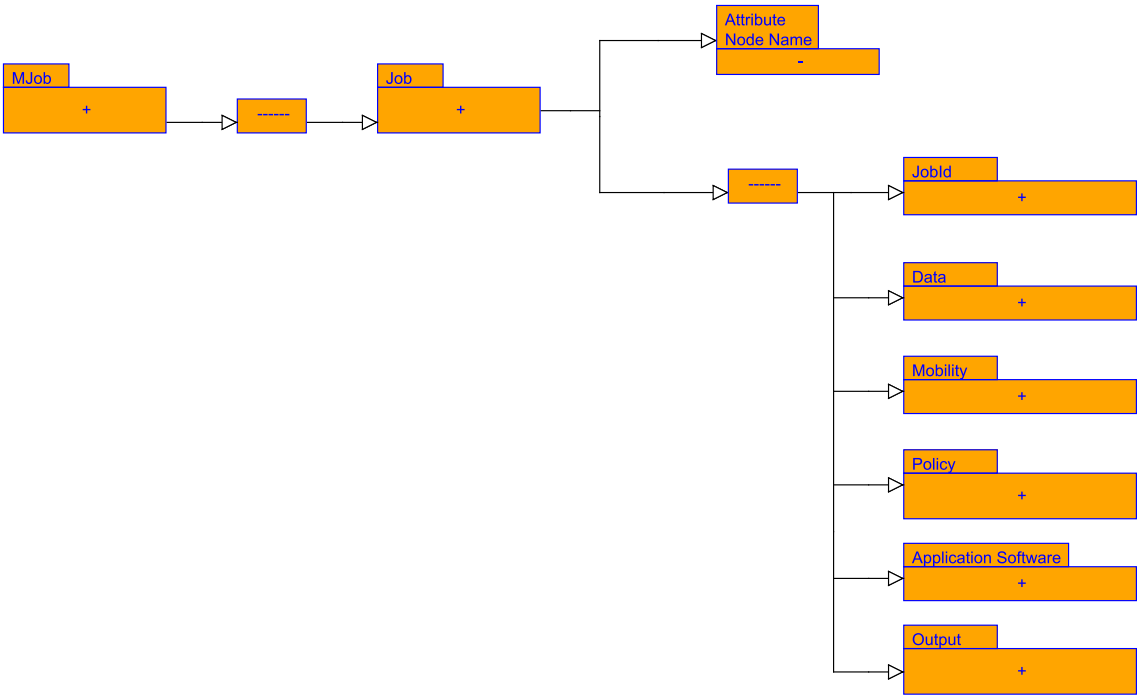

Figure 6.2: Internal-JSDL Schema

6.2.2 XML Encoding for Internal-JSDL Components

6.2.2.1 The main Component (MJob)

This is the main component of Internal-JSDL. The multi-divergence of this component is one. It includes Job components that have one or more job(s), every one of them has its own dedicated node.

6.2.2.2 Job Element

This component defines the requirements that are needed by the job to be operated on a grid resource(s). The multi-divergence of this component is one or more. It includes the JobId, Data, Mobility, Policy, Application Software and Output components.

- **Attributes**

  NodeName defines the node's name dedicated by the broker to operate the job. Its category is xsd:NCName.

- **Pseudo Schema**

```
<Job NodeName =‘xsd:ID”?/>
                <JobId.../>?
                <Data.../>+
                <Mobility.../>?
                <Policy.../>?
                <Application Software.../>+
                <Output.../>*
</Job>+
```

6.3 Job Identification (JobId)

In grid environments several jobs are required to accomplish the maximum utilization of grid resources. Therefore, each job should have a described name or identification to make it easy for the grid to manipulate each one of them.

6.3.1 Job Name

This component is a string that states the job name determined by the resource broker. The multi-divergence of this component is one.

- **Pseudo Schema**

```
<JobId>
        <JobName></JobName>
</JobId>
```

6.4 Data

It is a component that describes the data that should be available in the host that is going to execute the job prior to the job's execution starts. As shown in Figure (6.3). The multi-divergence of this component is one or more.

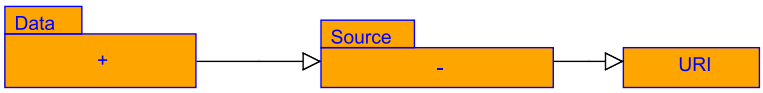

Figure 6.3: Data Schema

- **Pseudo Schema**

```
<Data>
        <Source>
</Data>*
```

6.4.1 Source

It includes the site location on the remote system of the directory or file that should be made available to the host that is holding the execution from the location described by the URI (Uniform Resource Identifiers) before starting the job. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<Source>
        <URI></URI>
</Source>?
```

6.4.1.1 URI

Uniform Resource Identifier describs the site location of the file or directory that should be made avalible for the job execution. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<Source>
        <URI>xsd:anyURI</URI>?
</Source>?
```

6.5 Mobility

This component is compromised of three elements, Application Software, Data and Running Job. The multi-divergence of the mobility component is zero or one. Figure (6.4) shows Mobilty components.

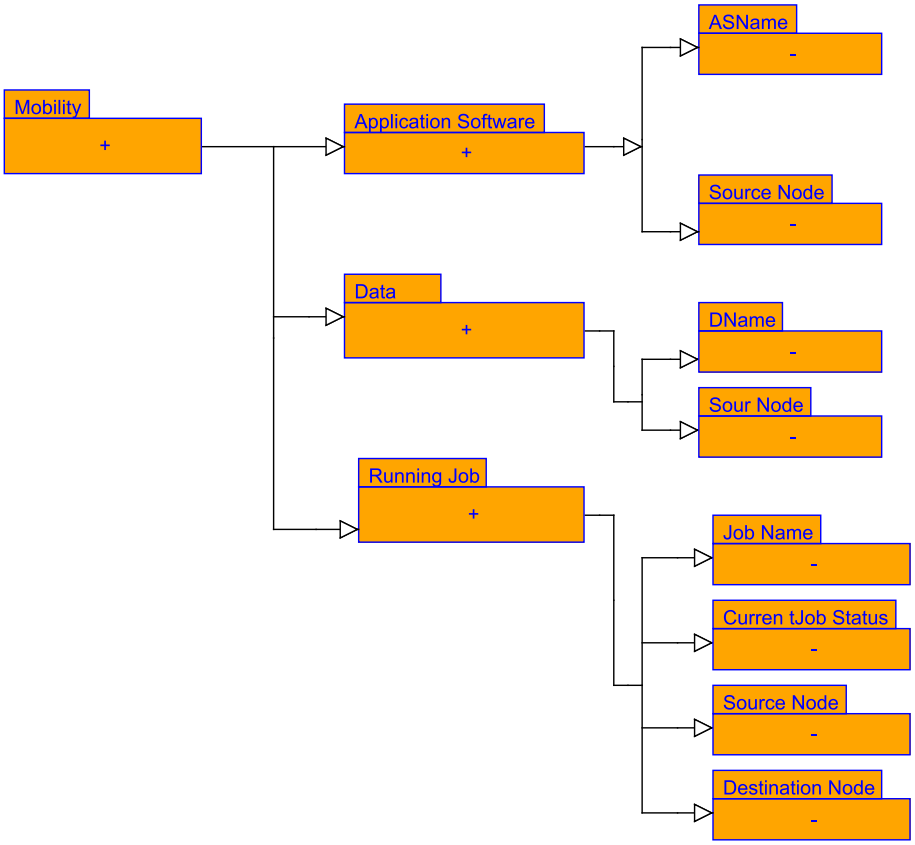

Figure 6.4: Mobility Schema

- **Pseudo Schema**

```
<Mobility>
        <ApplicationSoftware.../>?
        <Data...>*
        <RunningJob.../>*
</Mobility>?
```

6.5.1 Application Software

The application software component develops the node to have the ability to fit the job specifications if the node does not have the needed application software. This

component defines the mobility and its conditions for application software by having all components needed to mobile the application software between grid nodes. If this element is not defined, there is no migration for the application software and the node possibly has the needed application software. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<ApplicationSoftware>
        <ASName/>
        <SourceNode/>
</ApplicationSoftware>?
```

6.5.1.1 The Application Name (ASName)

This component is a string that states the name of the needed application software. If this element is not defined, there is no migration for application software and the node possibly has the needed application software. The multi-divergence of this component is one.

- **Attributes**

  (MobilityType) is an attribute that defines the mobility type. Application software mobility is divided into two types: copy or move. “Copy” means to obtain the application software and remain a copy of this application software on the resource. “Move” means to obtain the application software without saving a copy in the resource. The resource broker decides this attribute according to node policy.

- **Pseudo Schema**

```
<ASName MobilityType=“copy”|“move”>xsd:string</ASName>
```

6.5.1.2 Source Node

This component is a string that states the name of the node that owns the needed application software and its permits mobility. The resource broker decides this component according to node policy and availability of the needed application software. The multi-divergence of this component is one.

- **Pseudo Schema**

```
<SourceNode> xsd:string </SourceNode>
```

6.5.2 Data

This component includes all the components required to mobile data between grid nodes. If this element is not defined, there is no migration for data because either the job does not need the data or the node already owns it. The multi-divergence of this component is zero or more.

- **Pseudo Schema**

```
<Data>
        <DName/>
        <SourceNode/>
</Data>*
```

6.5.2.1 Data Name (DName)

This component is a string that determines the name of the needed data with amulti-divergence of one.

- **Attributes**

  Data mobility is divided into two types: copy or move. “Copy” means to obtain

the data and remain a copy of it on the resource. “Move” means to obtain the data without saving a copy in the resource. The resource broker decides this attribute according to node policy.

- **Pseudo Schema**

<DName MobilityType=“copy”|“move”>xsd:string</DNamee>

6.5.2.2 Source Node

This component is a string that states the name of the node that owns the needed data and permits its mobility. The resource broker decides this component according to node policy (by asking the mobile policy agent) and availability of the needed data (by asking the replica catalogue). The multi-divergence of this component is one.

- **Pseudo Schema**

<SourceNode> xsd:string </SourceNode>

6.5.3 Running Job

This component includes all the needed components to mobile the job from the needed node to other nodes that fulfill the running job conditions in a grid system. If this component is not defined, there is no mobility for the current job. The multi-divergence of this component is zero or more.

- **Pseudo Schema**

```
<RunningJob>
        <JobName/>
        <CurrentJobStatus>?
        <SourceNode/>
        <DestinationNode/>
</RunningJob>*
```

6.5.3.1 Job Name

This component is a string that defines the job name determined by a resource broker with a multi-divergence of one.

- **Pseudo Schema**

```
<JobName>xsd:string</JobName>
```

6.5.3.2 Current Job Status

This component is defined as a string including the status of job execution in order to start on a new node. The execution state comprises of program counter, running code, stored processor registers, local variables and return addresses. The reasons behind moving the current job between grid resources are partial or full failure or load balance. The resource broker defines this component for node failure, but in the evolution's case, the grid nodes are responsible for this definition. The multi-divergence of this component is zero or one .

- **Pseudo Schema**

```
<CurrentJobStatus>xsd:string</CurrentJobStatus>
```

6.5.3.3 Source Node

This element is a string that states the name of the node that is running the job and has the ability to migrate it. The resource broker decides this component according to node policy and availability of the needed data. The multi-divergence of this component is one.

- **Pseudo Schema**

```
<SourceNode>xsd:string</SourceNode>
```

6.5.3.4 Destiniation Node

This element is a string that states the name of the node that can accept the running job and resume it. The resource broker decides this component according to node policy and availability of the needed data. The multi-divergence of this component is one.

- **Pseudo Schema**

```
<DestiniationNode>xsd:string</DestiniationNode>
```

6.6 Policy

This decides how a job should be done, how security is applied in a domain and how an organization organizes, secures and distributes its resources. The multi-divergence of this component is zero or one. It has manay components as shown in Figure (6.5) followed by an explanation for each one of these components.

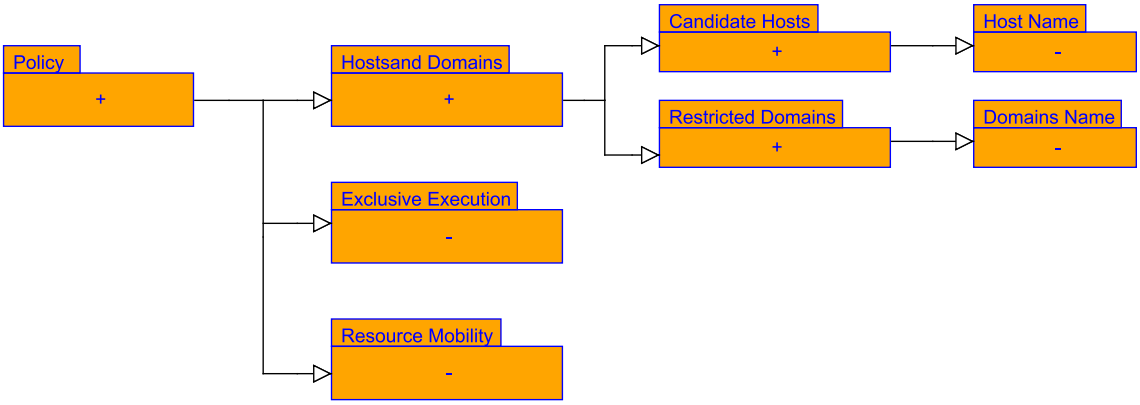

Figure 6.5: Policy Schema

- **Pseudo Schema**

```
<Policy>
        <HostsandDomains/>?
        <ExclusiveExecution>?
        <ResourceMobility/>?
</Policy>?
```

6.6.1 Hosts and Domains

It states the name of the hosts that may be chosen to execute the user's job and the name of the domains that are not allowed to execute or mobile the user's job in it. The multi-divergence of this component is zero or one .

- **Pseudo Schema**

```
<HostsandDomains>
        <CandidateHosts/>
        <RestrictedDomains/>
</HostsandDomains>?
```

6.6.1.1 Candidate Hosts

This states the name of the hosts that may be chosen to execute the user's job. It has to support the HostName component. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<CandidateHosts>
        <HostName></HostName>
</CandidateHosts>
```

* **HostName** is a string with a multi-divergence of one or more. It states the host names. It can be either a node name or logical set of hosts(cluster). The multi-divergence of this component is one.

- **Pseudo Schema**

```
<HostName>xsd:string</HostName>
```

6.6.1.2 Restricted Domains

This is a compound component with a multiplicity of zero or one. It states the name of the domains that are not allowed to execute or mobile the user's job in it. It has to support the DomainName component. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<RestrictedDomains>
        <DomainName></DomainName>
</RestrictedDomains>
```

* **DomainName** is a string with a multi-divergence of one or more. It states the domains' names. It can be either a domain name or logical set of domains (cluster). The multi-divergence of this component is one.

- **Pseudo Schema**

```
<RestrictedDomains>xsd:string</RestrictedDomains>
```

6.6.2 Exclusive Execution

This component is a Boolean type with a multi-divergence of zero or one that determines if the job allows only itself to run at any times on the allocated resources or not. In the "True" option, the job should be executed on the chosen resource exclusively. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<ExclusiveExecution>xsd:boolean</ExclusiveExecution>?
```

6.6.3 Resource Mobility

This component is a Boolean with a multi-divergence of zero or one that determines if the resource that is executing the job allows moving (mobile) this job to other resources. In the true choice, the job should have the ability to mobile between hosts according to the CandidateHosts and DomainName components. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<ResourceMobility>xsd:boolean</ResourceMobility>?
```

6.7 Application Software

To execute a job on a specific resource, this resource should have the appropriate application software. If the user has not specified that application, the Internal-JSDL document identifies it as a null job. To identify a software application, it is essential to specify the application's software name and its version. In some cases the difference versions of software applications does not affect the job's execution. This element has a multi-divergence of zero or one defining the application and its conditions. It includes two components as shown in Figure (6.6).

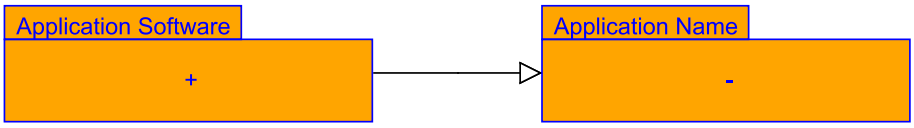

Figure 6.6: Application Software Schema

- **Pseudo Schema**

```
<CurrentApplicationSoftware>
                <ApplicationSoftwareName.../>
</CurrentApplicationSoftware>?
```

6.7.1 Application Software Name

This is a string component with a multi-divergence of one; it is used to determine the application's name.

- **Pseudo Schema**

```
<ApplicationSoftwareName>xsd:string</ApplicationSoftwareName>?
```

6.8 Output

This component is responsible for determining the files that should be transferred (staged out) from the host that is holding the job execution. As shown in Figure (6.7). The multi-divergence of this component is one or more. Files are staged out after the job terminates.

Figure 6.7: Output Schema

- **Pseudo Schema**

```
<Output>
                <Destination>
</put>*
```

6.8.1 Destination

This component is a compound component with a multi-divergence of one; it includes the location of the directory or file that should be staged out to from the location described by the URI when the job finishes.

- **Pseudo Schema**

```
<Destination>
                <URI></URI>
</Destination>?
```

6.8.1.1 URI

Uniform Resource Identifier describs the site location of the file or directory that should be made avalible to stage out a resulted file from the job execution. The multi-divergence of this component is zero or one.

- **Pseudo Schema**

```
<Source>
        <URI>xsd:anyURI</URI>?
</Source>?
```

6.9 Summary

The language that has been introduced in this chapter and in the previous chapter can be considered one of our major contributions in this thesis. In this chapter we have defined an Internal-JSDL language that has been modified to be a suitable language for communication between resource brokers and resources, as well as communications between resources themselves. The advantages of this language are supporting the mobility feature for jobs, data and application software and policy feature which give both the grid users and grid resource the ability to control and protect their jobs and resources in the grid environment. In this chapter we discussed the components of this language along with the XML code. We introduced in section two an overview of our language followed by five sections for expressing (jobs, data, mobility, policy and application software) in this new language. Finally the last section introduced the output that is responsible for determining the files that should be transferred (staged out) from the host that is holding the job execution.

Chapter 7

Simulation

7.1 Introduction

This chapter provides the simulation for mobile grid environment to evaluate our framework algorithm. This chapter is organized as follow: The first section introduces the description of the simulation followed by a description of how to configure grid environment. Section Three describes the node configurations; these configurations include hardware specifications, data, application software and policies. And the last section describes the job configurations including hardware specifications, data, application software and policies.

7.2 Simulation

In our simulation design we built a heterogeneous grid environment which has an unlimited number of resources in a fully connected topology. These nodes have the ability to migrate data, application software and jobs between them. The migration depends on the grid, resources policies and grid users' policies. We have developed a Java user interface that can simplify our work by creating a grid environment, configuring its nodes by each with its own application software, data, policies, hardware specifications, node names and finally creating an interface that can help the grid users to send

their jobs to the grid system.

The grid system has been simulated by using Jade simulator, which is a software framework fully implemented in Java language and allows agents to execute tasks defined according to the agent policy.

7.2.1 Simulation Description

When running the simulation, the main portal interface turns up. It composes all the functions needed to configure a new grid with all of its elements as shown in Figure (7.1). In this interface, the grid administrator is able to create a new grid environment by choosing grid name, configuring grid nodes, and sending jobs to the grid system. The interface will then directly pass all this information to the Jade simulator to create them.

The interfaces allows the grid, nodes and jobs requirements to be described simply, at which point the system will convert these requirements into Internal and External JSDL language and send them it to the Jade system, at the same time our system produces an XML file that describes the actions needed by the grid administrator and users in order to retrieve the system later. Once the Jade receives these files, it performs all the necessary grid functions. These functions include sending jobs to the suitable resources and migrate Jobs, application software's and/or data if required.

7.3 Grid Configuration

Using the Interface shown in Figure (7.1) the grid administrator can create a new grid environment by choosing the grid name and any domains and/or users who are not allowed to work under this grid. This can be done by entering the names of these domains and/or users in the Restricted Domain and Names field. This field will be translated into XML file once the administrator clicks on the create botton, and to be sent later to Jade simulator to create the grid environment under this policy.

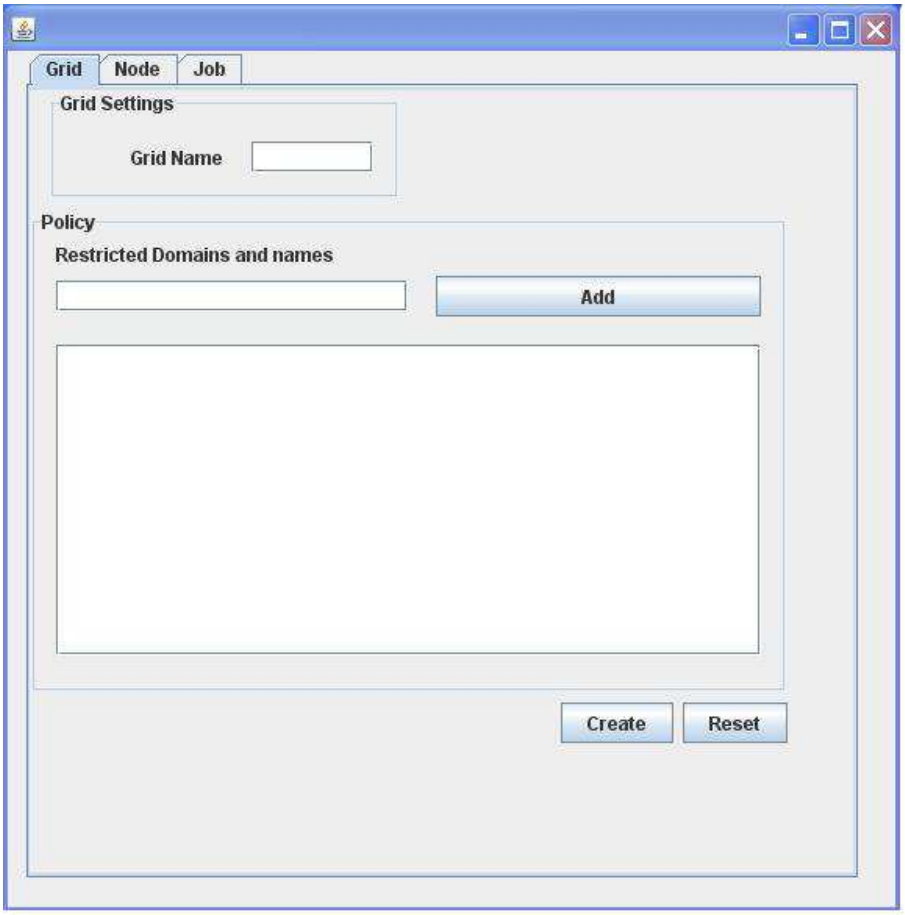

Figure 7.1: Main Simulation Interface

7.4 Node Configuration

Our interface can simulate nodes by configuring their specifications. This is done by specifying their names, grid names, domain name, number of jobs that can be processed at the same time, hardware specifications, application software, data and policies, as shown in Figure (7.2).

After determining the node's name, the administrator chooses the domain name for each node. This domain name helps in sorting out the nodes into groups, which in turn helps in making later the policy decisions. The administrator also has the ability to determine how many jobs each node can handle at the same time. The other fields are described as follow:

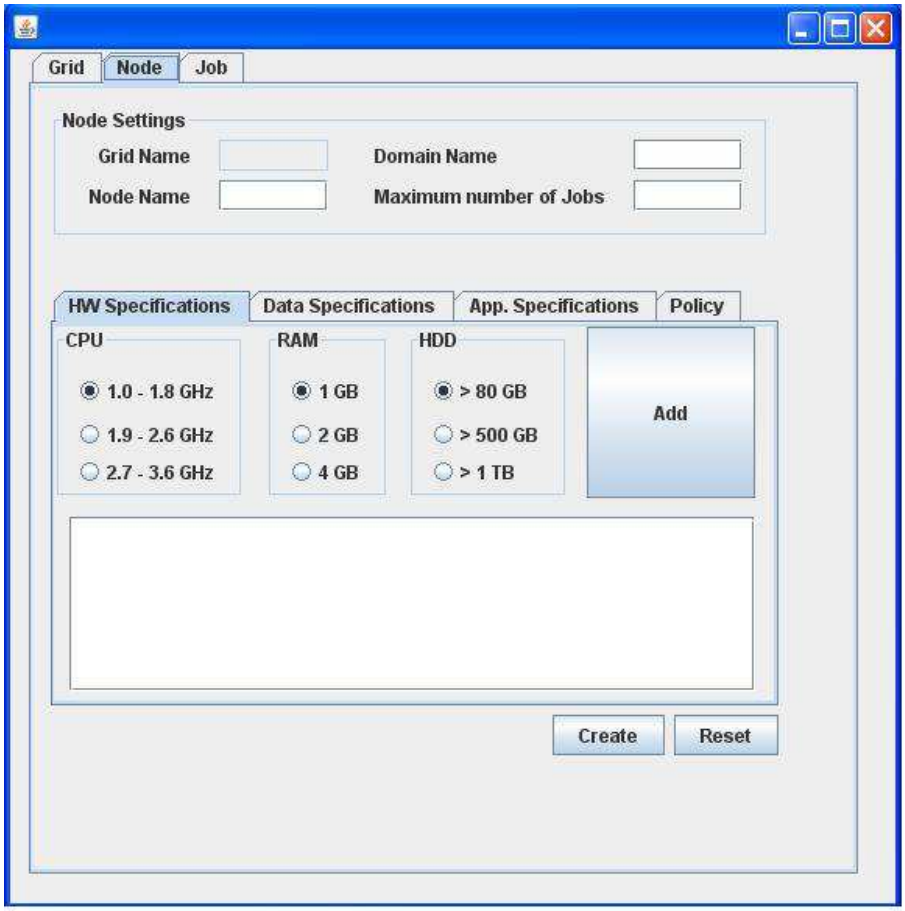

Figure 7.2: Node Configuration Interface

7.4.1 Node Hardware Specification

In this step the administrator determines the node's hardware specifications. These specifications include CPU speed, memory size and hard disk space as shown in Figure (7.2). By choosing these specifications the system represents these choices by an agent which is understood by the Jade simulator and at the same time store them into the hardware section in an XML file which represents the overall node's specification.

7.4.2 Node Data Specification

The interface shown in Figure (7.3) helps the administrator to configure the data in the node by determining the data name and policy. If the administrator chooses a "Moving"

feature when creating a new data, the policy for this single data will allow this data to be moved wherever it is allowed to be moved. Otherwise it will prevent it from moving from its original node. The “Copy” feature has the same job but it will perform a copy action for the data instead of moving.

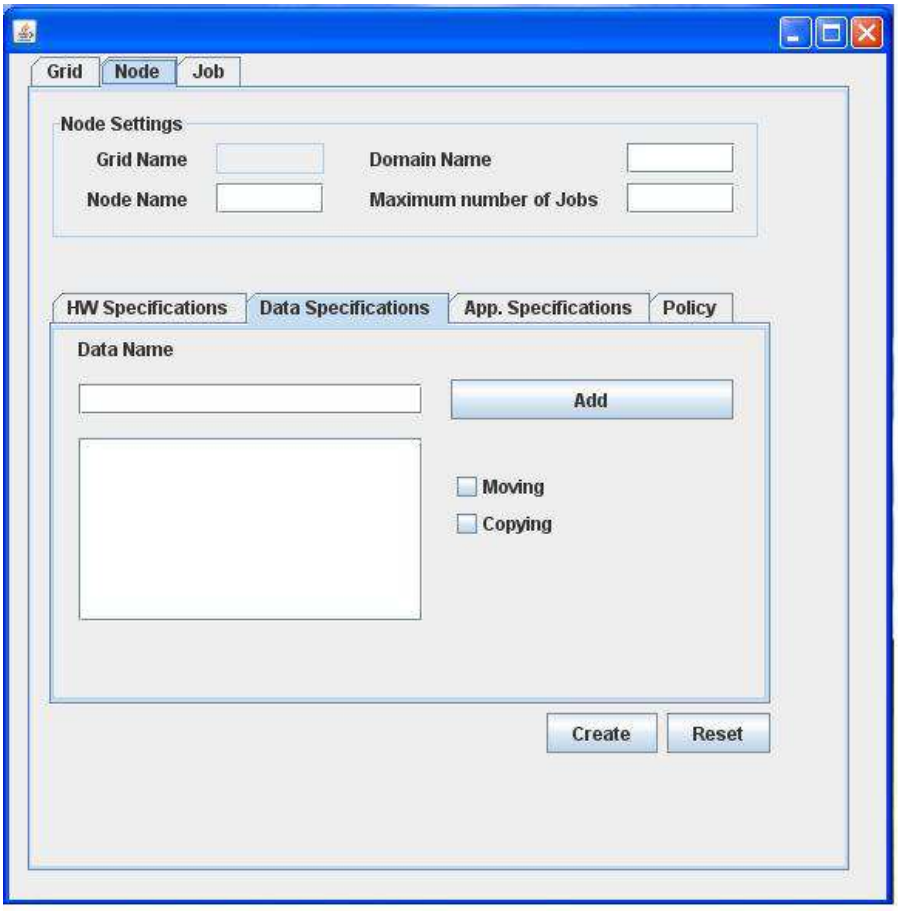

Figure 7.3: Data Configuration Interface for a Single Node

After finishing creating the node‘s data, the system represents each single data item by an agent which can be understood by the Jade simulator and at the same time store the policy for this single data into the data section in the XML file which represents the overall node’s specification.

7.4.3 Node Application Software Specification

The interface shown in Figure (7.4) helps the administrator to configure the Application software's in each node by determining the application name and the policy for this application. If the administrator chooses a "Moving" feature when creating new application software, the policy for this single application will allow this application to be moved wherever it is allowed to be moved. Otherwise it will prevent it from moving from its original node. The "Copy" feature has the same job but it will perform a copy action for the application instead of moving.

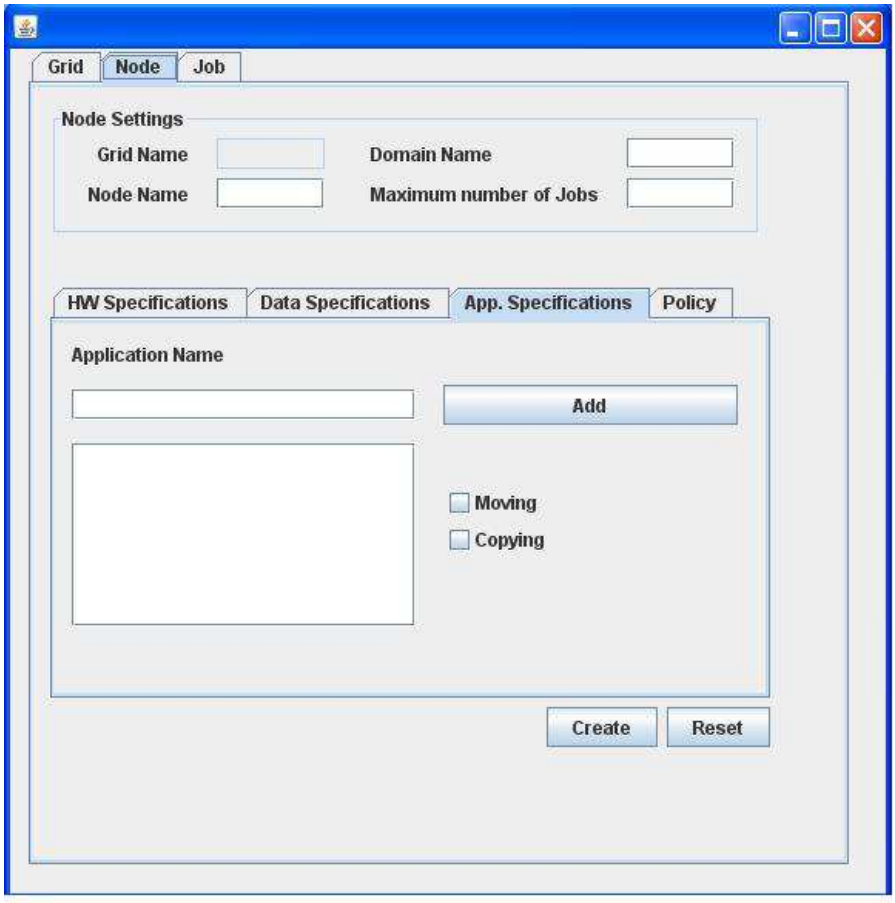

Figure 7.4: Application Software Configuration Interface for a Single Node

After finishing from creating node's application software, the system represents each single application by an agent which can be understood by the Jade simulator and at

the same time stores the policy for this single application in the application software section in the XML file which represents the overall node's specification.

7.4.4 Node Policy Specification

The interface shown in Figure (7.5) helps the administrator to configure the policy of ech node by determining any restricted domain(s) or user(s) with whom they are not allowed to work under this node. The administrator can determine whether this node is allowed to execute two jobs (or more) at the same time or not. This feature can be applied using the "Exclusive" choice in the policy. By choosing this option, the node is not allowed to execute more than one job at the same time.

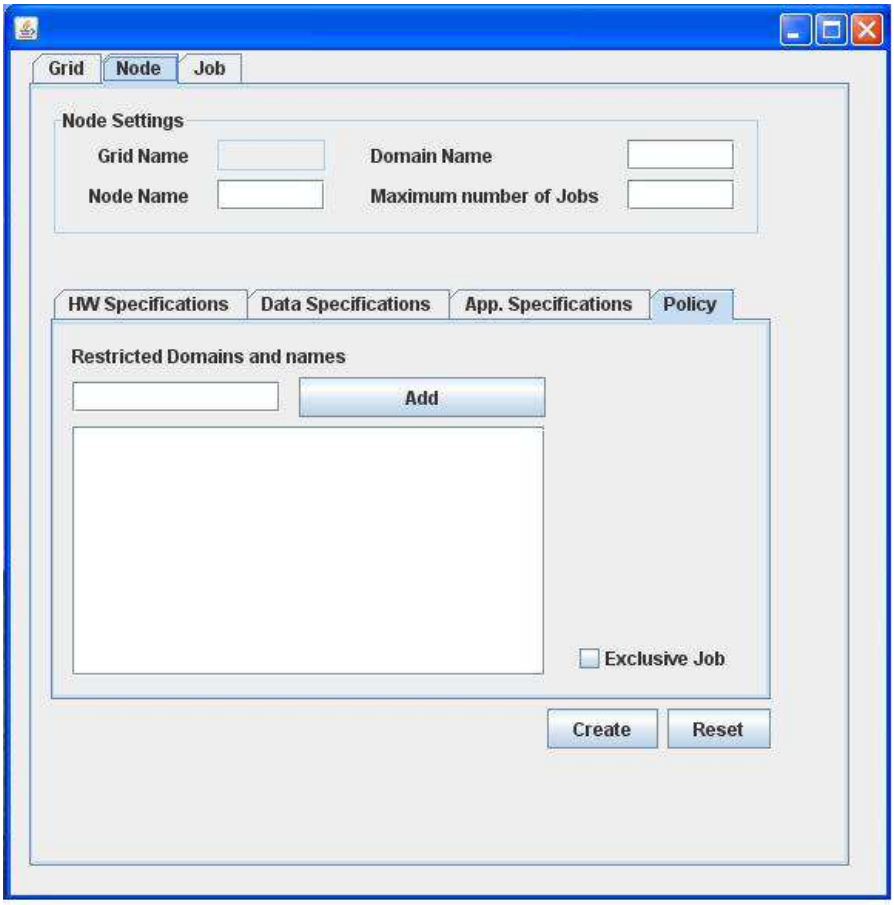

Figure 7.5: Policy Configuration Interface for a Single Node

After creating node‘s policy, the system will store the policy for this node into the policy section in the XML file which represents the overall node’s specification.

7.5 Job Configuration

Figure (7.6) shows interface that helps users describe their jobs requirements in a simple way. These requirements are then be converted by the system to a language that can be understood by the Jade simulator, and at the same time it is going to be stored in an XML file that describes the jobs with their policies.

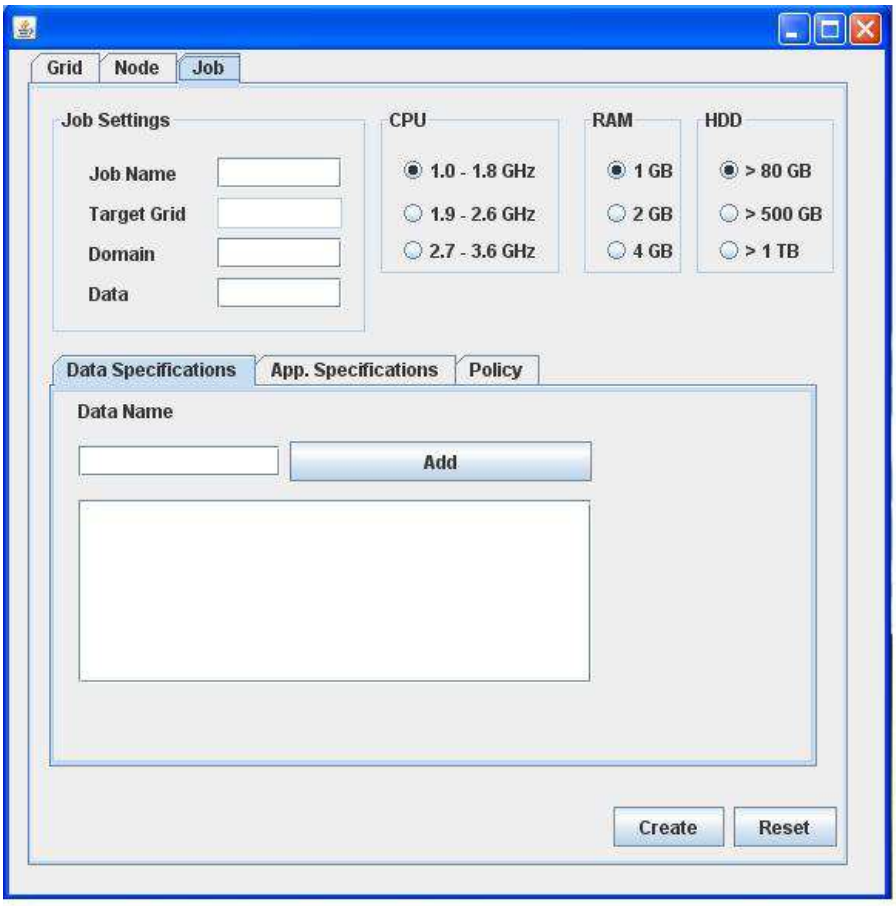

Figure 7.6: Job Configuration Interface

Our interface can accept jobs by configuring their requirements. This is can be done by specifying their names, grid names, domain name, any data attached with the job,

hardware specifications, application software, data and policies, as shown in Figure (7.6).

After determining the job's name, the administrator can choose the domain name for that job. This domain name helps in sorting out the jobs into groups, which will help in sending the jobs to the appropriate domain later. The administrator also has the ability to attach a specific data along with the job to be processed during the execution time to complete the job. The other fields are described as follow:

7.5.1 Job Hardware Specification

In this step the administrator can determine the job's hardware specifications. These specifications include CPU speed, memory size and hard disk space. By choosing these specifications the Interface will represent these choices by an agent which can be understood by the Jade simulator and at the same time store them into the hardware section in an XML file which represents the overall job's specification. As shown in Figure (7.6).

7.5.2 Job Data Specification

The interface shown in Figure (7.6) helps the administrator to configure the data needed to process the job by the grid nodes. By determining this name the system will add this data to the job requirements which will be sent later to the Jade simulator to find the suitable node(s) that owns this data. At the same time the system will store the name of this data in the XML file that describes the job requirements.

7.5.3 Job Application Software Specification

The interface shown in Figure (7.7) assists the administrator to choice the application software(s) required to handle the job by the grid nodes. By choosing this name(s) the system will attach it to the job requirements which will be posted later to the Jade simulator to locate the suitable node(s) that have this application(s). At the same time the system will save the name of this application(s) in XML file that expresses the job

requirements.

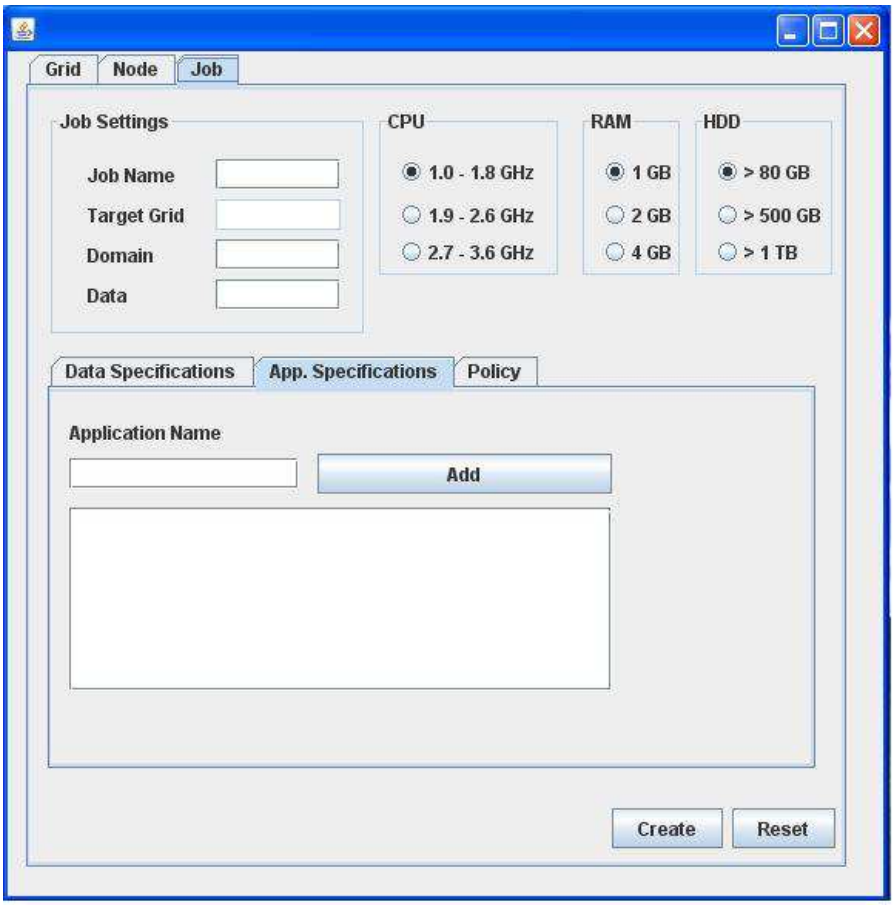

Figure 7.7: Application Software Configuration Interface for Single Job

7.5.4 Job Policy Specification

The interface shown in Figure (7.8) helps the administrator to configure the job's policy before submitting it to the grid environment. The administrator or the grid users can determine any restricted domain(s) or user(s) whith whom they are not allowed to handle their jobs. Also in this section the administrator determines whether this job is allowed to be executed with other jobs at the same time or not. This feature can be applied using the Exclusive choice in the policy. By choosing this option, the job is not allowed to execute with other jobs. The Moving feature allows the job to be moved

wherever it is allowed to be moved. Otherwise it will prevent it from moving from one node to another. By choosing this feature the system will store the mobility feature in the policy section in the XML file that presents the job's specifications.

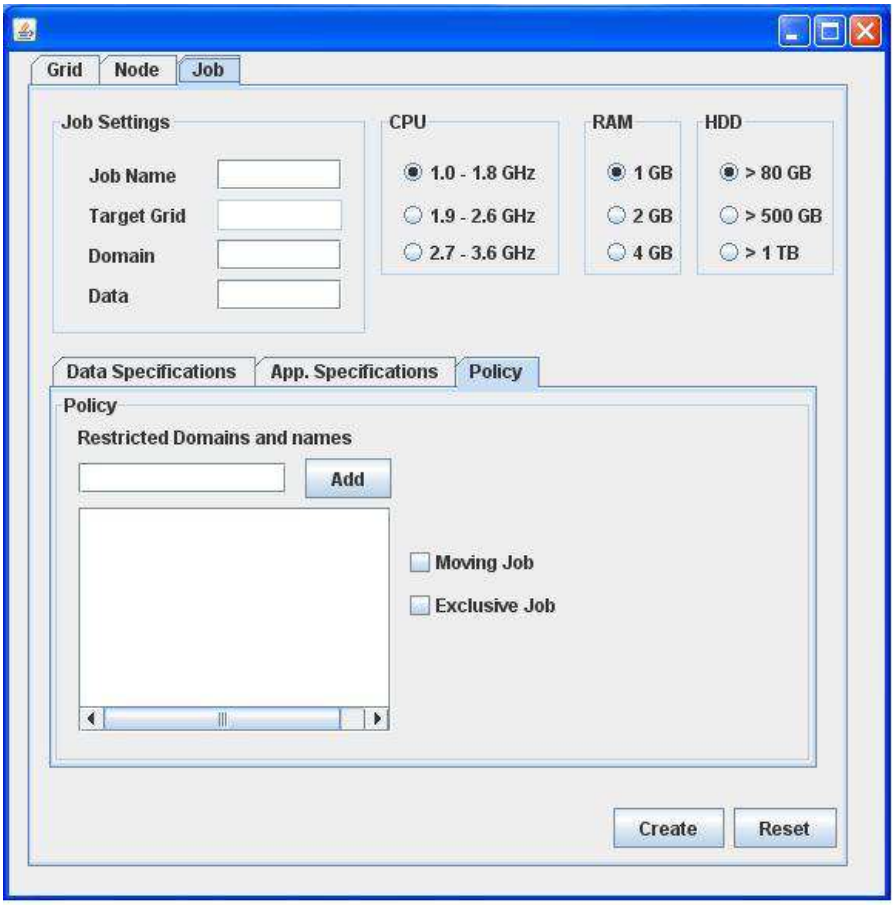

Figure 7.8: Policy Configuration Interface for a Single Job

After creating job's policy, the system will store the policy for this job into the policy section in the XML file which represents the overall job's specification.

7.6 Summary

We presented our simulation for the grid environment in the case of applying policies over mobility. It illustrated how the simulation is constructed and worked as in section three. And how to configure nodes and jobs as in sections four and five.

Chapter 8

Simulation Validation and Evaluation

8.1 Introduction

In the previous chapter we introduced our simulation and the way to help the grid administrators to configure the grid environment along with it nodes. Also we introduced our interfaces that help the grid users to configure and describe their jobs. In this chapter we are going to validate our simulation. Section two introduces three scenarios the first one describes the way to configure grid environment. The second and third scenarios describe nodes and jobs configurations. In our scenarios we have configured grid nodes in different domains under different administrators, and then we applied the resource sharing mobility and finally we evaluate the ability of our framework policies to control these mobility according to the users and node administrators rules. The second section shows the evaluation and the results of the simulation depending on number of rejected jobs and the overall used nodes in the system without applying the policies over mobility as a first case then applying the policies over mobility in the grid system as a second case.

We are going to proof that applying the policy rules on the top of mobility gives the

grid, grid's nodes and grid's users the ability and the privacy to control over their data, application software's and jobs and at the same time distributing jobs to most of the grid's nodes instead of utilizing just partial of grids nodes. Also we are going to proof our research contributions including supporting a multi-organization environment with different domains, providing a clear support for sharing mobile resources between multiple heterogeneous VOs, support the user preferences in its final decision and enforcing data policies in its designs.

8.2 Simulation Validation

The aim of this part is to validate the output of the user interfaces matches the input of the Jade Simulator and the XML files that are responsible for retrieving the grid environment later. Three scenarios are used to judge the simulation.

8.2.1 First Scenario: Grid Configuration

In the first scenario we created a grid environment called Test-1 by using the interface shown in Figure (7.1). This interface is divided into two parts: the first part creates the grid environment in Jade simulator as shown in Figure (8.1) and the XML file related to the grid policy which contains the restricted domain(s) and user(s) as shown in Figure (8.2).

8.2.2 Second Scenario: Node Configuration

In the second scenario we created four nodes in the Test-1 grid environment by using the interfaces in Figures (7.2- 7.5). The nodes have been named as following UK-A1, UK-A2, USA-A3 and Chian-A4 related to their domain names. The results of these interfaces are divided into two parts: the first one adding the new nodes to Test-1 grid environment in Jade simulator as shown in Figure (8.3) and the XML files related to policy for each node as shown in Figure (8.4).

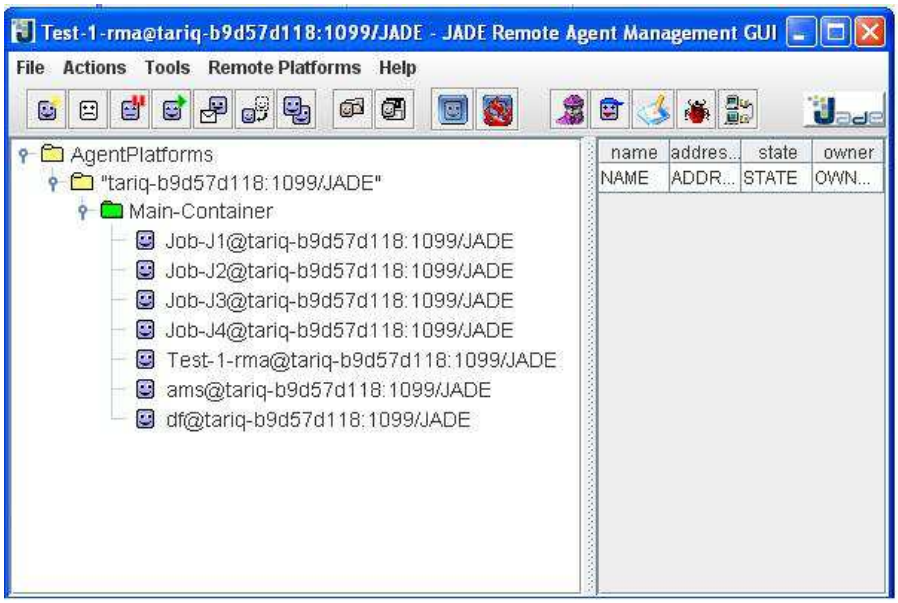

Figure 8.1: Screen-Shot of Test-1 Grid Environment

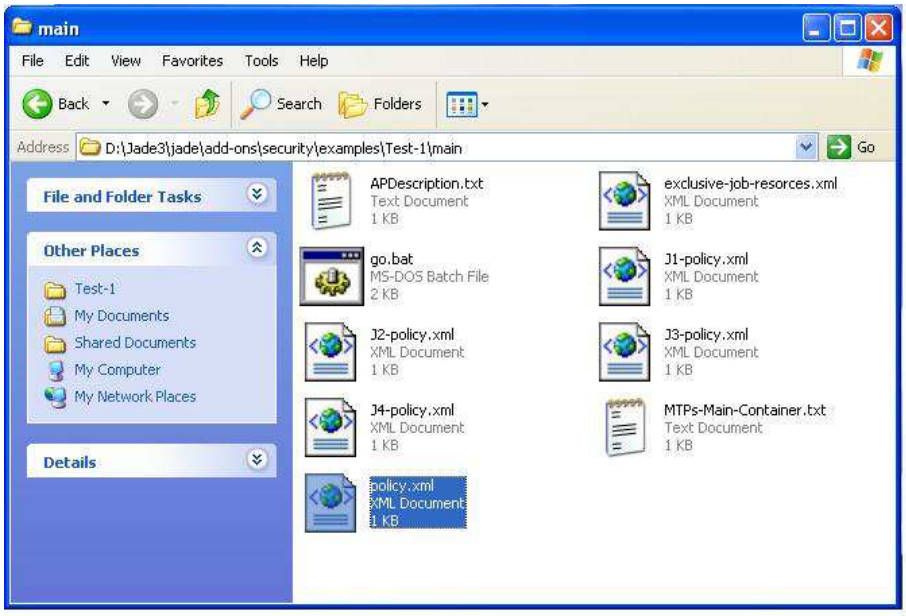

Figure 8.2: Screen-Shot of XML File for Test-1 Grid Policy

8.2.3 Third Scenario: Job Configuration

In the third scenario we submitted four jobs to the Test-1 grid environment by using the interfaces in Figures (7.6- 7.8). The jobs have been named as following J1, J2, J3

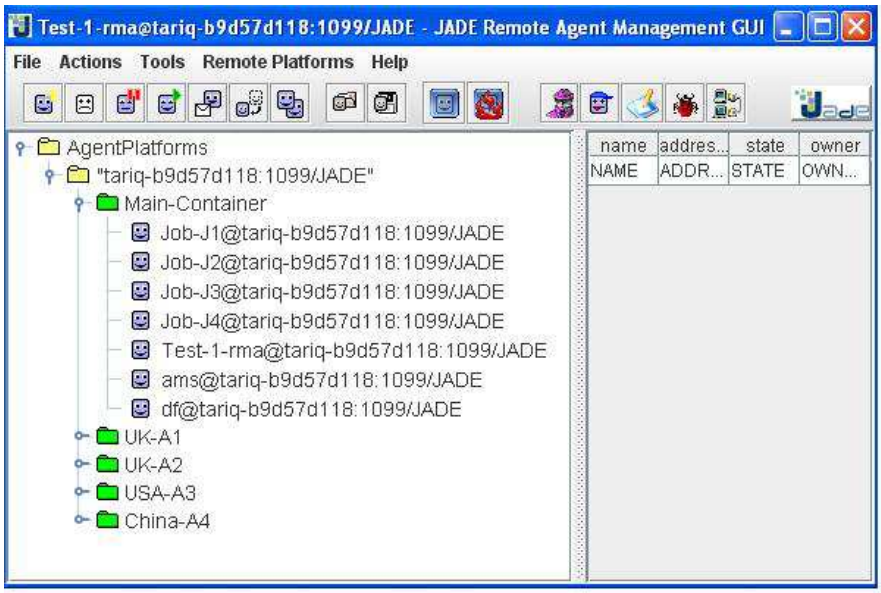

Figure 8.3: Screen-Shot of Test-1 Grid Environment showing four Nodes

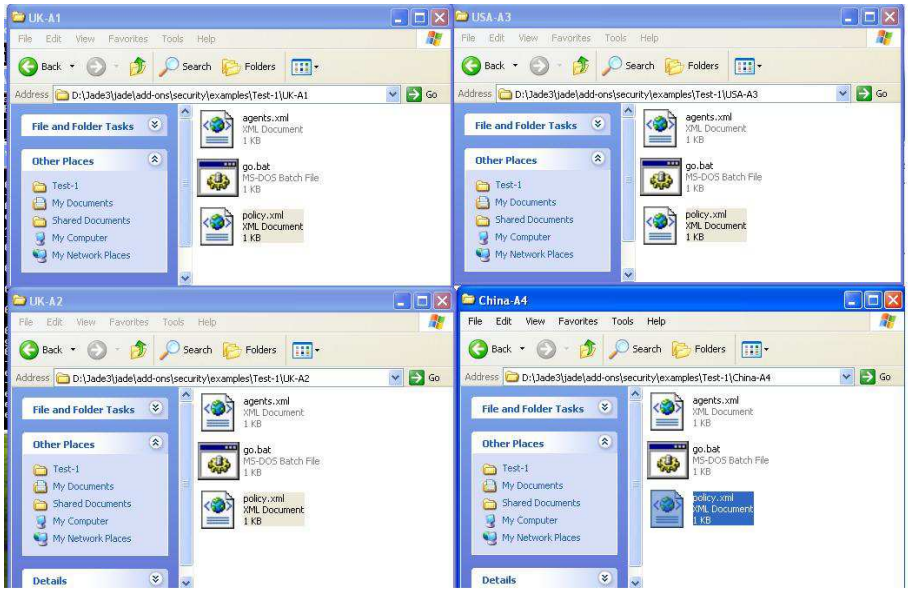

Figure 8.4: Screen-Shot of XML Files for Test-1 Node Policies

and J4. The results of these interfaces are divided into two parts: the first one adding the new jobs to Test-1 grid environment in Jade simulator as shown in Figure (8.3) and the XML files related to the policies for each job as shown in Figure (8.5).

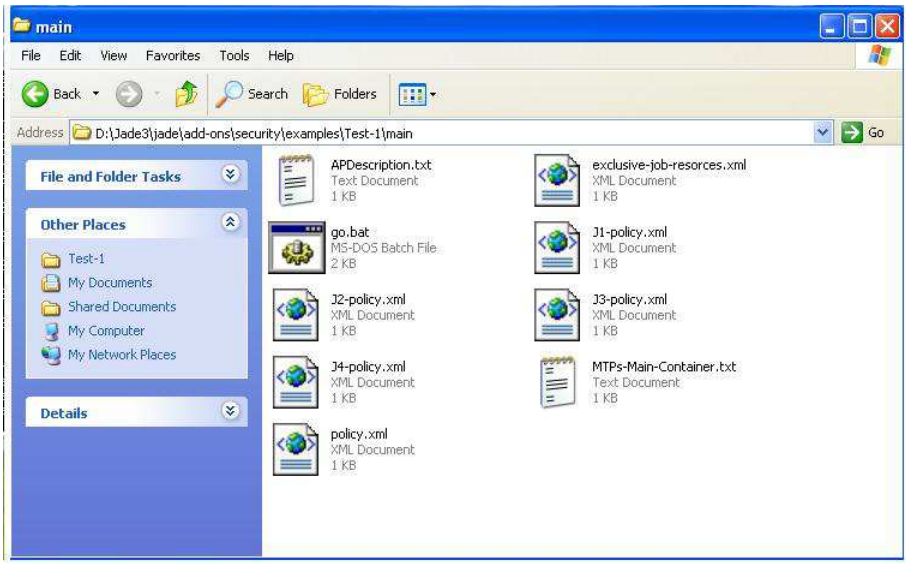

Figure 8.5: Screen-Shot of XML Files for Test-1 Job Policies

8.3 Evaluation

We applied specific grid environments in our simulation with specific nodes and jobs. Each one of them with different hardware specifications, data, application software and polices. Our aim is to simulate and analyze the effect of the policy on the resource mobility (jobs, data and application software) in the grid environment.

Our program lets job, data and application software migrate from one node to another in the grid environment. The aim of the simulation is to present the effect of the policies on the number of rejected jobs and number of nodes used in the grid during these migrations. To accomplish these aims, we constructed a grid environment that contains 20 nodes; each node has distinctive (or similar) hardware, application software and data specifications from others. Afterward we sent 30 jobs sequentially to this environment. The reason behind choosing these numbers is to build a simple system to obtain straight results that can be understood by the reader and to show the impact of the policies on the system. After building the system we applied the job and resource mobility within the grid according to the following scenarios and configurations:

- Case 1: No mobility. We configured the policies for all of the jobs, data and

application software not to be allowed to migrate within the grid along with preventing grid's nodes to accept migration resources between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. The Figures from (8.6 to 8.13) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of no mobility.

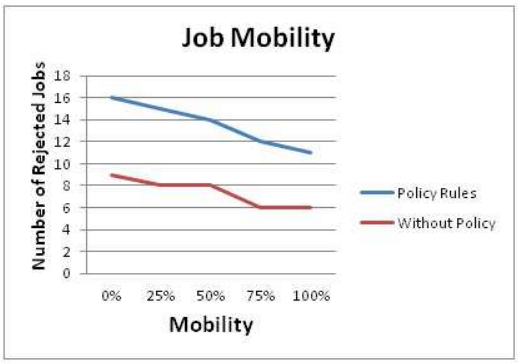

Figure 8.6: Rejected Jobs with Job Mobility

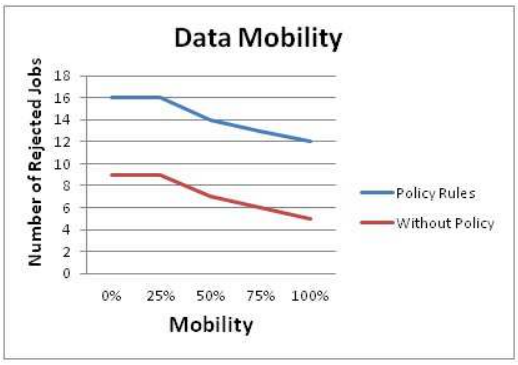

Figure 8.7: Rejected Jobs with Data Mobility

- Case 2: Partial Mobility (%25). We configured quarter of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. Also we configured quarter of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to ac-

complish these jobs. The Figures from (8.6 to 8.13) show the effects of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of Partial Mobility (%25).

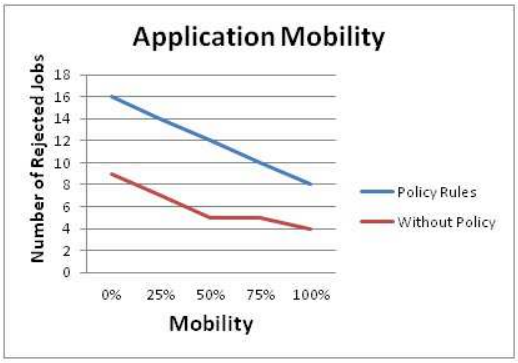

Figure 8.8: Rejected Jobs with Application Software Mobility

- Case 3: Partial Mobility (%50). We configured half of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. We configured half of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs.

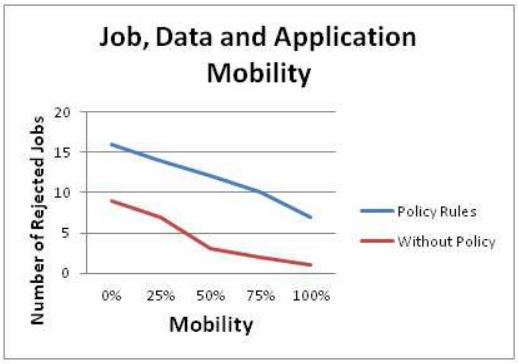

Figure 8.9: Rejected Jobs with Job, Data and Application Software Mobility

- Case 4: Partial Mobility (%75). We configured (%75) of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. We configured (%75) of the grid's nodes polices to accept resource migra-

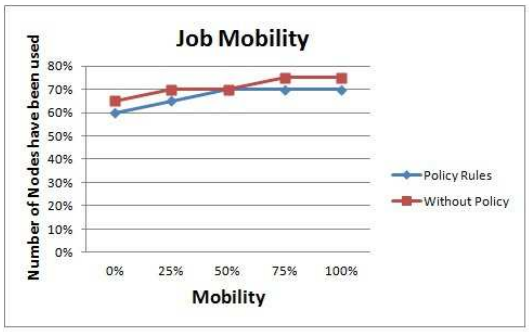

Figure 8.10: Overall Used Nodes with Job Mobility

Figure 8.11: Overall Used Nodes with Data Mobility

tion between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures from (8.6 to 8.13) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of Partial Mobility (%75).

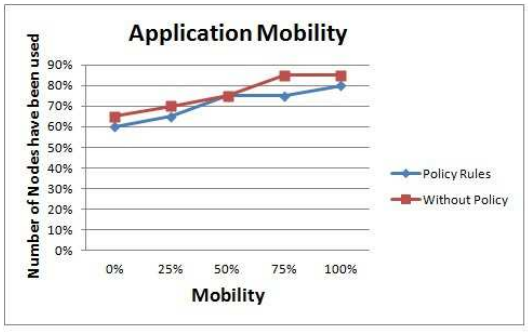

Figure 8.12: Overall Used Nodes with Application Software Mobility

- Case 5: Full Mobility. We configured all of the policies for the jobs, nodes, data and application software to be allowed to migrate within the grid. We configured all of the grid's nodes polices to accept resource migration between them. We then sent 30 jobs sequentially to the grid with distinctive (or similar) hardware, application software and data needed to accomplish these jobs. Figures from (8.6 to 8.13) show the effect of polices on number of rejected jobs and number of the overall nodes used in the grid in the case of full Mobility.

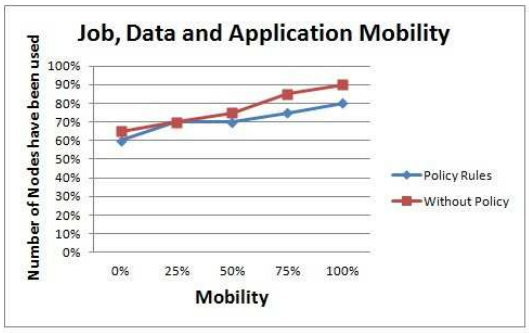

Figure 8.13: The overall Used Nodes with Job,Data and Application Software Mobility

The previous figures show that applying the mobility solution has solved the lack of finding suitable resources to fulfil grid users jobs. As it can be seen clearly when the percentage of the mobility increased, number of rejected jobs is decreased and at the same time distributing jobs to most of the grid's nodes instead of utilizing just partial of grids nodes has increased too. The figures show that in the case of applying the mobility policies rules; the number of rejected jobs is more than without applying the policies rules; as each administrator and users applying their own policies and preferences. Nevertheless, applying the policy rules on the top of mobility gives the grid, grid's nodes and grid's users the ability and the privacy to control over their data, application software's and jobs.

From the previous experiments, we have answered our research question that says "How does the grid interact with policies for different domains and organizations in the case of mobile sharing and data movements?" and at the same time, we answered the other research questions that asks "How to introduce policy management tools that

provides support for sharing mobile resources between multiple heterogeneous VOs?" and "How to design a policy framework that can support the user policy in its final decision?"

8.3.1 Rejected jobs

Once the grid is not able to accept a job due to the short of available resources, amount of rejected jobs rises. Mobility has solves this problem by migrating data or application software needed by the new jobs or even evacuated the required node by moving the running job to fit the new jobs if necessary. Our results show the effect of the mobility on number of rejected jobs when it's applied. This means that the grid can fit a resource to execute the job, consequently; the number of rejected jobs in the grid will be reduced. Nevertheless; that reducing is affected by the grid, node and job policies. When applying these policies, the number of rejected jobs is less than the situation when the policies are not applied (mobility by itself).

To make it clear, we applied our experiments into stages. In the first stage we applied only the job migrations to see the effects on number of rejected jobs as in Figure (8.6). In the second stage only data migrations were applied as in Figure (8.7). The next stage the application software migrations were applied as show in the results in Figure (8.8). And finally, all types of migrations were applied as in Figure (8.9).

It can be seen clearly that number of rejected jobs when applying the policies is less than without polices, but applying the policies over mobility gives the grid, grid's nodes and grid's user's the ability and the privacy to control over their data, application software's and jobs.

8.3.2 Overall Nodes Usage

In the normal case, not all the grid's nodes own data or application software needed by all the users' jobs. In this case that some nodes are not utilized due to the short in these resources, and most of the jobs are sent to a specific nodes because of the fact that these nodes own the needed resources required by most of the jobs than the other nodes, or in some cases the level of security and the policy conditions for some jobs

or nodes are higher than others. In these cases the gird finds itself in a situation not to accept a job, at some point, due to the short of the available resources. Mobility solves this problem by migrating data or application software needed by the new jobs, or even evacuated the required node to fit the new jobs if necessary. In this case the mobility helps in distributing user's jobs to most of the grid's nodes, and that's will help in load balancing and reducing number of rejected jobs. Our results show the effect of the mobility on number of nodes used by the grid to fulfill the grid user's jobs. In the first stage we applied only the job migrations to see how many number of nodes had been used to execute jobs sent to the grid as in Figure (8.10). In the second stage only data migrations were applied as in Figure (8.11). The next stage the application software migrations were applied as show in the results in Figure (8.12). And finally, all types of migrations were applied as in Figure (8.13).

As a result, when applying the mobility solution, more nodes had been used than the situation without mobility. Moreover, applying the policy rules on the top of that gives the grid, grid's nodes and grid's user's the ability and the privacy to control over their data, application software's and jobs, and at the same time distributing jobs to most of the grid's nodes.

8.4 Summary

In this chapter we presented the validation for our simulation. We applied three scenarios to configure grid environment, grid nodes and grid jobs as in sections two. In section three we evaluated our scenarios by simulating number of rejected jobs and the overall used nodes in the system without applying the policies over mobility as a first case then applying the policies over mobility in the grid system as a second case.

The evaluation and case studies show that applying the mobility solution has solved the lack of finding suitable resources to fulfil grid users' jobs. When the percentage of the mobility increased, number of rejected jobs is decreased and at the same time distributing jobs to most of the grid's nodes instead of utilizing just partial of grids' nodes has increased too.

Also, the evaluation shows that in the case of applying the mobility policies rules; the number of rejected jobs is more than without applying the policies rules; as each administrator and users applying their own policies and preferences. Nevertheless, applying the policy rules on the top of mobility gives the grid, grid's nodes and grid's users the ability and the privacy to control over their data, application software's and jobs.

Furthermore, we have used the External-JSDL to define users grid jobs and preferences by presenting their identifications, application software, resource specifications, data, user data and policies that are needed to accomplish the user's job. And later the Internal-JSDL has been used as a language that is used to communicate between the resource broker and the grid nodes, and between the grid nodes themselves in different organizations and domain.

As a result, we have answered our research question that says "How does the grid interact with policies for different domains and organizations in the case of mobile sharing and data movements?" and at the same time, we answered the other research questions that asks "How to introduce policy management tools that provides support for sharing mobile resources between multiple heterogeneous VOs?" and "How to design a policy framework that can support the user policy in its final decision?"

Based on the evaluation and case studies all research questions have been answered. We have proofed our research contributions including supporting a multi-organization environment with different domains, providing a clear support for sharing mobile resources between multiple heterogeneous VOs, support the user preferences in its final decision and enforcing data policies in its designs.

Chapter 9

Conclusions and Future Work

9.1 Summary

Grid computing appears as a result of a combination of multi-network computer system to develop a wide range and heterogeneous system used to solve scientific or industrial problems. It should have the ability to organize resources, which are not under the subject of centralized domain, utilize protocols and interfaces and supply high quality of service. As a result of such technology many challenges such as finding suitable resources, reducing number of rejected jobs and providing privacy and security stand in front of developing such technology.

Grid computing technologies is used as low-cost systems to congregate and utilize computational resources together. Grids was initially created on the idea that resources infrastructure are dynamic and heterogeneous in their nature. That's mean different organization with different administrative domains. That's also mean that security was considered from the beginning when the grid system initially built.

Mobility is the ability to migrate or relocate jobs, data and application software among grid nodes. It facilitates the accomplishment of requirements for grid jobs as well as grid users. It also assists grid evolution, improves performance of operating applications by relocating data to the target host, therefore reducing the communication consumption and solving the load balancing issues. These Mobility (or migrations)

depend on the grid's users and the grid's nodes policies.

Policies are groups of regulations, standards and practices written by the administrators of resources about how their resources or jobs can be handled and used. Every resource applies its own security policy that may result in the refusal of requests for utilizing of its resources. Grid computing has solved a lot of large and complex problems that require interaction and cooperation between different resources and jobs, but these resources belong to different domains and administrators. Each job has different requirements and specification in order to be executed in the grid. To facilitate the ability to improve mobile resource sharing between multiple heterogeneous VOs, a policy management framework is needed to support the heterogeneity in the policy frameworks in different domains under different administrators.

Another aspect that should be taken into account is the user preferences, before users can submit their jobs or run their applications on a certain source or system they may need to guarantee that this source or system has not been compromised which could result in their own application or data being stolen. To date, not enough attention has been paid to policies that deal with such concerns. Most existing grid systems support only limited types of policies (e.g. CPU resources) [44], [60],[81].

Grid security tools has several security challenges, one of them is the data confidentiality, which is the security of data movement within the network from intruders' attacks [86]. Data confidentiality can be addressed by limiting the mobility to trusted parts of the grid, but this solution leads to the notion of Virtual Organizations (VOs). So we have proposed a way to limit mobility by using policies. But when policy checking for the resources and jobs takes place in computational grids, the heterogeneity, diversity of policies and attributes leads to a need for policy management tools that can handle these heterogeneity and diversity.

In this research, a new policy management tools have been introduced to solve the mobility limitation and data confidentiality especially in the case of mobile sharing and data movements within the grid. We presented a dynamic and heterogeneous policy management framework that can give a clear policy definition about the ability to move jobs, data and application software from nodes to nodes during jobs' execution

in the grid environment. This framework supports a multi-organization environment with different domains, supports the external grid user preferences along with enforces policies for data movements and the mobility feature within different domains.
The results of our research have been implemented using the Jade simulator, which is a software framework fully implemented in Java language and allows agents to execute tasks defined according to the agent policy. The simulation results have verified that the research aims enhance the security and privacy performance in the grid environments, enhance control over data and services distribution and usage and present practical evidence in the form of scenario test-bed data as to the effectiveness of our architecture.

9.2 Contributions

Traditional authorization policy management frameworks act well in authorization policy for a single Virtual Organization (VO) where the contributing hosts grant the permission to follow a global authorization system. We have presented a dynamic and heterogeneous policy management framework that can supports moving jobs, data and application software from node to node during dynamic jobs' execution in multi VOs in grid environments.
This new framework for policy management has the ability to supports a multi-organization environment with different domains. Our contribution in this section is proposing an extension to that framework in [121] to be able to provide the features of supporting the grid user preferences along with enforcing policies for data movements and resource mobility feature within different domains under different administrators. This extension has been published in Communications in the Computer and Information Science Conference (CoNeCo2011) [15] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110]. The extension provides the new features of supporting the external grid users' preferences along with enforcing policies for data movements within different domains. The framework also provides a clear support for sharing mobile resources between multiple heterogeneous VOs. With the assist of the

mobility, the services can move over the grid so as to obtain data from grid nodes, implement their jobs on those nodes and carry the results back to their original nodes. Our framework provides privacy and security to support the mobility by using the mobile policy agent which plays a significant role to achieve this privacy and security. This contribution has been introduced in chapters three and four and has been published in Risk and Security of Internet and Systems Conference (CRiSIS), IEEE Computer Society, 2011 [14] and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

This research also supports user preferences in the final policy decisions. Our framework provides a guarantee that the user jobs or data are not going to be compromised, by preventing their own application or data from being stolen. Our contribution in this section where proposed as an extension to that framework in [121] so as to be able to provide the features of supporting the grid user preferences in the final decision before applying the resource mobility feature. This contribution has been introduced in chapters three and four and has been presently published in the Communications in Computer and Information Science Conference (CoNeCo2011), the Risk and Security of Internet and Systems Conference (CRiSIS2011) and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

Data policy has a role within our framework. Our framework enforces data policies within grid environments. Therefore, the grid administrator and the grid users have more control over their data movements within the grid, especially during the mobility operation itself. The administrators or the users can determine the domains and users who are allowed to access and use their data. Handling and store this data are also being controlled by our system. This and has been presently published in the Communications in Computer and Information Science Conference (CoNeCo2011), the Risk and Security of Internet and Systems Conference (CRiSIS2011) and the International Journal of Computer Networks and Communications (IJCNC), 2012 [110].

We presented our new language that supports and expresses new policy framework. Our grid system has been simulated by using Jade simulator, which is a software framework fully implemented in the Java language that allows agents to execute tasks defined

according to the agent policy. In order to build the proposed grid system we designed interfaces that help grid administrators build their grid. At the same time these interfaces give grid users the ability to describe and send their jobs to the grid. The language that is used to design these interfaces is the Java language. When grid administrators or grid users submit their requirements by using the previous interfaces to be simulated by Jade, our system converts these requirements to an External-JSDL that expresses the jobs requirements and this is understood by the grid environment no matter what domain the resources are. Our tool converts these requirements to an Internal-JSDL; as a language that is used to communicate between the resource broker and the grid nodes, and between the grid nodes themselves in different organizations and domains. Then the system stores these language expressions as an XML schema in order to be retrieve later and sends them to Jade to be simulated. The reason for XML as a language for grid and job requirements expression is that XML has many attractive attributes such as the simplicity in reading, understanding and processing by users and computers. The External-JSDL language has been introduced in chapter five and the Internal-JSDL language has been introduced in chapter six.
All of our contributions have been tested, simulated and fully evaluated in a pure grid environment that supports the mobile sharing and data movements.

9.3 Future Work

This thesis has introduced a new framework represents a step in ongoing research efforts to reach an efficient grid environment. Many issues still need investigation. The most clear are as follows:

- The framework can be use to support a market models by allowing the grid to use the cheapest domains and resources available. The fast growing in grid and cloud computing has put them as next generation computing platforms. They enable the foundation of virtual organization for sharing resources that is distributed across the world. The resource holder of each of these resources has several usage, policies and cost paradigm. However, different cost and method in each

domain and organization make it difficult for the grid users to choose the best and cheapest resource so as to accomplish their jobs. We aim to develop our framework so as to have the ability to identify the ideal domains that can support the users jobs and at the same time keep the privacy of the users and the resource owners data and resources.

- We have introduced a modern XML based language to express our framework. Our target is to develop our ideas towards mapping languages for grid and or integration of our policy language and combine it with other modern languages. In our thesis we have introduce a new language that can support the grid users' jobs (External-JSDL) and the communications between the resource broker and grid nodes and between nodes themselves (Internal-JSDL) whatever their domains or administrators. The feature of this language can be combined with features of other languages that support the grid services such as Language Resources which talk about using and sharing language resources, Language Grid for Communication which is interested in using the language for intercultural collaboration and Language Grid for Translation which is using the language for Wikipedia translation and localization [70].

  Developing our language gives the opportunity to take advantage of services offered by other technology in grid environment and the opportunity to make our applications available to others as a solution to express the policy in grid systems. Also to make this language easy to use and learn, allow complicated tasks to be executed in relatively few stages and make the editing and operating of the code faster than what we have.

- The addition of authorisation and authentication to our system would provide a more complete security and privacy level. We mentioned in our architecture that authorisation and authentication are not offered in our framework at this level as it is not a key part of the simulation carried out to test the efficiency of our architectures. An execution approach to include these to the framework is of interest.

Bibliography

[1] The datagrid project, http://eu-datagrid.web.cern.ch/eu-datagrid/

[2] The globus resource specification language rsl v1.0, http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html

[3] Gridpp, the grid and industry, www.gridpp.ac.uk/posters/superseded/GridPP.ppt

[4] Gridpp, uk computing for particle physicists, http://www.gridpp.ac.uk/

[5] Job scheduling hierachically (josh). Website, http://gridengine.sunsource.net/josh.html

[6] A. Anjomshoaa, F.B., Michel Drescher, D.F., An Ly, S. McGough, D.P., Savva, A.: Job submission description language (jsdl) specification, version 1.0 (November 2005), http://www.gridforum.org/documents/GFD.56.pdf

[7] A. Litke, D. Skoutas, T.V.: Mobile grid computing: changes and challenges of resourse management in a mobile grid environment. In: 5th International Conference on Practical Aspects of Knowledge Management (2004)

[8] Abramson, D., Buyya, R., Giddy, J.: A computational economy for grid computing and its implementation in the nimrod-g resource broker. Future Gener.

Comput. Syst. 18, 1061–1074 (October 2002), http://portal.acm.org/citation.cfm?id=765837.765844

[9] Aldabbas, O.: A framework for mobility and temporal dimensions of grid system. Ph.D. thesis, School of Computing, De Montfort University, UK (2008)

[10] Alhusaini, A.H., Prasanna, V.K., Raghavendra, C.S.: A unified resource scheduling framework for heterogeneous computing environments. In: Proceedings of the Eighth Heterogeneous Computing Workshop. p. 156. HCW '99, IEEE Computer Society, Washington, DC, USA (1999), http://portal.acm.org/citation.cfm?id=795690.797903

[11] Ali Anjomshoaa, Fred Brisard, M.D.D.F.A.L.S.M.D.P., Savva, A.: Job submission description language (jsdl) specification, version 1.0 (November 2005), http://www.gridforum.org/documents/GFD.56.pdf

[12] Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., Foster, I.: The globus striped gridftp framework and server. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing. p. 54. SC '05, IEEE Computer Society, Washington, DC, USA (2005), http://dx.doi.org/10.1109/SC.2005.72

[13] Alliance, G.: Globus toolkits, http://www.globus.org

[14] Alwada'n, T., Janicke, H., Aldabbas, O., Alfawair, M.: New framework for policy support for mobile grid services. In: Cuppens, F., Foley, S., Groza, B., Minea, M. (eds.) CRiSIS. pp. 88–93. IEEE (2011), http://dblp.uni-trier.de/db/conf/crisis/crisis2011.html#AlwadanJAA11

[15] Alwada'n, T., Janicke, H., Aldabbas, O., Hamza, A.: New framework for dynamic policy management in grid environments. In: Abdulkadir Ozcan, Jan Zizka, D.N. (ed.) Recent Trends in Wireless and Mobile Networks. pp. 297–304. Springer Berlin Heidelberg (2011)

[16] Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. Web Semant. 5(2), 58–71 (Jun 2007), http://dx.doi.org/10.1016/j.websem.2007.03.002

[17] Athanaileas, T.E., Tselikas, N.D., Tsoulos, G.V., Kaklamani, D.I.: An agent-based framework for integrating mobility into grid services. In: Proceedings of the 1st International Conference on Mobile Wireless MiddleWARE, Operating Systems, and Applications. pp. 31:1–31:6. MOBILWARE '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium (2007), http://portal.acm.org/citation.cfm?id=1361492.1361531

[18] Attiya, H., Welch, J.: Distributed computing: fundamentals, simulations, and advanced topics. John Wiley and Sons, Inc., Hoboken, New Jersey, USA, second edn. (2004)

[19] Azzedin, F., Maheswaran, M.: Integrating trust into grid resource management systems. In: Proceedings of the 2002 International Conference on Parallel Processing. p. 47. ICPP '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=850943.853031

[20] Azzedin, F. Maheswaran, M.: Evolving and managing trust in grid computing systems. Electrical and Computer Engineering, IEEE CCECE 2002 3, 1424 (August 2002)

[21] B. Jacob, M. Brown, K.F., Trivedi, N.: Introduction to grid computing. IBM Corp., Riverton, NJ, USA (2005)

[22] Bajaj, S., et al.: Web services policy framework (ws-policy). In: BEA, IBM, Microsoft and SAP (2004)

[23] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., Wright, T.: Transport layer security (tls) extensions (2003)

[24] Boozallen, INC, H.: Federal public key infrastructure x.509 certificate and crl extensions profile (October 2005), http://www.idmanagement.gov/fpkipa/documents/fpki_certificate_profile.pdf

[25] Buyya, R., Abramson, D., Giddy, J.: A case for economy grid architecture for service-oriented grid computing. In: Proceedings of the 15th International Parallel & Distributed Processing Symposium. p. 83. IPDPS '01, IEEE Computer Society, Washington, DC, USA (2001), http://portal.acm.org/citation.cfm?id=645609.662300

[26] Buyya, R., Vazhkudai, S.: Compute power market: Towards a market-oriented grid. Cluster Computing and the Grid, IEEE International Symposium p. 574 (2001)

[27] Cao, J., Jarvis, S.A., Spooner, D.P., Turner, J.D., Kerbyson, D.J., Nudd, G.R.: Performance prediction technology for agent-based resource management in grid environments. In: Proceedings of the 16th International Parallel and Distributed Processing Symposium. p. 265. IPDPS '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=645610.661728

[28] Cardelli, L.: Secure internet programming. chap. Abstractions for mobile computations, pp. 51–94. Springer-Verlag, London, UK (1999), http://portal.acm.org/citation.cfm?id=380171.380179

[29] Carpenter, B.E., Janson, P.A.: Abstract interdomain security assertions: a basis for extra-grid virtual organizations. IBM Syst. J. 43, 689–701 (October 2004), http://dx.doi.org/10.1147/sj.434.0689

[30] Carzaniga, A., Picco, G.P., Vigna, G.: Designing distributed applications with mobile code paradigms. In: Proceedings of the 19th international conference on Software engineering. pp. 22–32. ICSE '97, ACM, New York, NY, USA (1997), http://doi.acm.org/10.1145/253228.253236

[31] Chakrabarti, A., Damodaran, A., Sengupta, S.: Grid computing security: A taxonomy. IEEE Security and Privacy 6, 44–51 (January 2008), http://portal.acm.org/citation.cfm?id=1344235.1344299

[32] Chunlin, L., Layuan, L.: An agent-based approach for grid computing. In: Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003. Proceedings of the Fourth International Conference on. p. 608. IEEE Computer Society, Washington, DC, USA (2003)

[33] Cody, E., Sharman, R., Rao, R.H., Upadhyaya, S.: Security in grid computing: A review and synthesis. Decis. Support Syst. 44, 749–764 (March 2008), http://dx.doi.org/10.1016/j.dss.2007.09.007

[34] Computing, I.G., Subramoniam, K., Maheswaran, M., Toulouse, M.: Towards a micro-economic model for resource allocation. In: In IEEE canadian conference on electrical and computer engineering. pp. 782–785. IEEE Press (2002)

[35] Coulouris, Dollimore, J., Kindberg, T.: Distributed Systems: Concepts and Design (4th Edition) (International Computer Science). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2005)

[36] Coulouris, G.F., Dollimore, J.: Distributed systems: concepts and design. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1988)

[37] David G. Rosado, E.F.M., Lopez, J.: Security services architecture for secure mobile grid systems. Systems Architecture (June 2010)

[38] Dierks, T., Rescorla, E.:

[39] Dimitrakos, T., Djordjevic, I., Matthews, B., Bicarregui, J., Phillips, C.: Policy-driven access control over a distributed firewall architecture. In: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02). pp. 228–. POLICY '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=863632.883510

[40] Düllmann, D., Segal, B.: Models for replica synchronisation and consistency in a data grid. In: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing. p. 67. IEEE Computer Society, Washington, DC, USA (2001), http://portal.acm.org/citation.cfm?id=874077.876514

[41] Dunlop, N., Indulska, J., Raymond, K.: Methods for conflict resolution in policy-based management systems. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing. p. 98. EDOC '03, IEEE Computer Society, Washington, DC, USA (2003), http://portal.acm.org/citation.cfm?id=942793.943139

[42] Ernemann, C., Hamscher, V., Schwiegelshohn, U., Yahyapour, R., Streit, A.: On advantages of grid computing for parallel job scheduling. In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. p. 39. CCGRID '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=872748.873249

[43] Ernemann, C., Hamscher, V., Yahyapour, R.: Economic scheduling in grid computing. In: Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing. pp. 128–152. JSSPP '02, Springer-Verlag, London, UK (2002), http://portal.acm.org/citation.cfm?id=646383.689702

[44] Feng, J., Cui, L., Wasson, G., Humphrey, M.: Policy-directed data movement in grids. In: Proceedings of the 12th International Conference on Parallel and Distributed Systems - Volume 1. pp. 319–326. ICPADS '06, IEEE Computer Society, Washington, DC, USA (2006), http://dx.doi.org/10.1109/ICPADS.2006.82

[45] Ferreira, L., Berstis, V., Armstrong, J., Kendzierski, M., Neukoetter, A., MasanobuTakagi, Bing, R., Amir, A., Murakawa, R., Hernandez, O., Magowan,

J., Bieberstein, N.: Introduction to grid computing with globus. IBM Corp., Riverton, NJ, USA, first edn. (2003)

[46] Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)

[47] Fitzgerald, S.: Grid information services grid information services for distributed resource sharing. In: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing. p. 181. IEEE Computer Society, Washington, DC, USA (2001), http://portal.acm.org/citation.cfm?id=874077.876489

[48] Fong, P.W.: Viewer's discretion: Host security in mobile code systems (1998), school of Computing Science, Simon Fraser University

[49] Forum, O.G.: Open grid forum, http://www.gridforum.org/

[50] Foster, I., Kesselman, K.: The grid: Blueprint for a future computing infrastructure. In: Morgan Kaufmann in Computer Architecture and Design (1999)

[51] Foster, I., Kesselman, K.: The Grid2: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann, second edn. (2003)

[52] Foster, I.: What is the grid? a three point checklist (June 2002), http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

[53] Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for computational grids. In: Proceedings of the 5th ACM conference on Computer and communications security. pp. 83–92. CCS '98, ACM, New York, NY, USA (1998), http://doi.acm.org/10.1145/288090.288111

[54] Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. 15(3), 200–222 (2001)

[55] Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D., Mirabile, F., Moore, L., Rust, B., Siegel, H.J.: Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In: Proceedings of the Seventh Heterogeneous Computing Workshop. p. 3. HCW '98, IEEE Computer Society, Washington, DC, USA (1998), http://portal.acm.org/citation.cfm?id=795689.797878

[56] Fuggetta, A., Picco, G.P., Vigna, G.: Understanding code mobility. IEEE Trans. Softw. Eng. 24, 342–361 (May 1998), http://dx.doi.org/10.1109/32.685258

[57] Galis, A., Gelas, J.P., Lefèvre, L., Yang, K.: Active network approach to grid management. In: Proceedings of the 2003 international conference on Computational science: PartIII. pp. 1103–1112. ICCS'03, Springer-Verlag, Berlin, Heidelberg (2003), http://portal.acm.org/citation.cfm?id=1762418.1762540

[58] Galis, A., Plattner, B., Smith, J.M., Denazis, S.G., Moeller, E., Guo, H., Klein, C., Serrat, J., Laarhuis, J., Karetsos, G.T., Todd, C.: A flexible ip active networks architecture. In: Proceedings of the Second International Working Conference on Active Networks. pp. 1–15. Springer-Verlag, London, UK (2000), http://portal.acm.org/citation.cfm?id=645638.663873

[59] Galstyan, A., Czajkowski, K., Lerman, K.: Resource allocation in the grid using reinforcement learning. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3. pp. 1314–1315. AAMAS '04, IEEE Computer Society, Washington, DC, USA (2004), http://dx.doi.org/10.1109/AAMAS.2004.232

[60] Grimshaw, A., Ferrari, A., Knabe, F., Humphrey, M.: Legion: An operating system for wide-area computing. Tech. rep., Charlottesville, USA (1999)

[61] Grimshaw, A.S., Humphrey, M.A., Natrajan, A.: A philosophical and technical comparison of legion and globus. IBM J. Res. Dev. 48, 233–254 (March 2004), http://dx.doi.org/10.1147/rd.482.0233

[62] Grimshaw, A.S., Humphrey, M.A., Natrajan, A.: A philosophical and technical comparison of legion and globus. IBM J. Res. Dev. 48, 233–254 (March 2004), http://dx.doi.org/10.1147/rd.482.0233

[63] Guan, T., Zaluska, E., De Roure, D.: A grid service infrastructure for mobile devices. In: Proceedings of the First International Conference on Semantics, Knowledge and Grid. pp. 42–. SKG '05, IEEE Computer Society, Washington, DC, USA (2005), http://dx.doi.org/10.1109/SKG.2005.10

[64] Heymann, E., Senar, M.A., Luque, E., Livny, M.: Adaptive scheduling for master-worker applications on the computational grid. In: Proceedings of the First IEEE/ACM International Workshop on Grid Computing. pp. 214–227. GRID '00, Springer-Verlag, London, UK (2000), http://portal.acm.org/citation.cfm?id=645440.652833

[65] Humphrey, M., Thompson, M.R.: Security implications of typical grid computing usage scenarios. Cluster Computing 5, 257–264 (July 2002), http://portal.acm.org/citation.cfm?id=592899.593008

[66] I. Foster, C. Kesselman, J.M.N., Tuecke, S.: The physiology of the grid: An open grid services architecture for distributed systems integration (2002), http://www.globus.org/alliance/publications/papers/ogsa.pdf

[67] Iamnitchi, A., Foster, I.T.: On fully decentralized resource discovery in grid environments. In: Proceedings of the Second International Workshop on Grid Computing. pp. 51–62. GRID '01, Springer-Verlag, London, UK (2001), http://portal.acm.org/citation.cfm?id=645441.652838

[68] Institute, I.S.: RFC 793-transmission control protocol (1981), http://rfc.sunsite.dk/rfc/rfc793.html, edited by Jon Postel. Available at http://rfc.sunsite.dk/rfc/rfc793.html

[69] Internet Engineering Task Force: RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification (September 1981), http://tools.ietf.org/html/rfc791

[70] Ishida, T.: The Language Grid: Service-Oriented Collective Intelligence for Language Resource Interoperability. Cognitive Technologies, Springer (2011), http://books.google.co.uk/books?id=VpUSjkk--0kC

[71] J., J.: Enablers for agile virtual enterprise integration. Agility and Global Competition (1(3)) (1997)

[72] Jacob, B.: Grid computing: What are the key components? In: ITSO Redbooks Project Leader, IBM (June 2003)

[73] Jacob, B., Ferreira, L., Bieberstein, N., Gilzean, C., Girard, J.Y., Strachowski, R., Yu, S.S.: Enabling applications for grid computing with globus. IBM Corp., Riverton, NJ, USA, first edn. (2003)

[74] Jameel, H., Kalim, U., Sajjad, A., Lee, S., Jeon, T.: Mobile-to-grid middleware: Bridging the gap between mobile and grid environments. In: EGC'05. pp. 932–941 (2005)

[75] Joshi, J.: Access-control language for multidomain environments. Internet Computing, IEEE 8(6), 40 – 50 (2004)

[76] K. Krauter, R.B., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Softw. Pract. Exper. 32, 135–164 (February 2002), http://portal.acm.org/citation.cfm?id=565293.565296

[77] Kaneda, K., Taura, K., Yonezawa, A.: Virtual private grid: a command shell for utilizing hundreds of machines efficiently. Future Gener. Comput. Syst. 19, 563–573 (May 2003), http://dx.doi.org/10.1016/S0167-739X(03)00036-0

[78] Kornblum, J.A., Raz, D., Shavitt, Y.: The active process interaction with its environment. Comput. Netw. 36, 21–34 (June 2001), http://portal.acm.org/citation.cfm?id=376733.376735

[79] Laszewski, G.V.: Grid computing: Enabling a vision for collaborative research. In: In Conference on Applied Parallel Computing, 3rd CSC Scientific Meeting, Lecture Notes. pp. 1–8. Springer (2002)

[80] Lorch, M., Adams, D.B., Kafura, D., Koneni, M.S.R., Rathi, A., Shah, S.: The prima system for privilege management, authorization and enforcement in grid environments. In: Proceedings of the 4th International Workshop on Grid Computing. p. 109. GRID '03, IEEE Computer Society, Washington, DC, USA (2003), http://portal.acm.org/citation.cfm?id=951948.952044

[81] Lorch, M., Kafura, D., Fisk, I., Keahey, K., Carcassi, G., Freeman, T., Peremutov, T., Rana, A.S.: Authorization and account management in the open science grid. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing. pp. 17–24. GRID '05, IEEE Computer Society, Washington, DC, USA (2005), http://dx.doi.org/10.1109/GRID.2005.1542719

[82] M. Humphrey, M.R. Thompson, K.J.: Security for grid. In: Proceedings of IEEE (3). vol. 93, p. 644 (March 2005)

[83] Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: Proceedings of the Eighth Heterogeneous Computing Workshop. p. 30. HCW '99, IEEE Computer Society, Washington, DC, USA (1999), http://portal.acm.org/citation.cfm?id=795690.797893

[84] Maheswaran, M., Krauter, K.: A parameter-based approach to resource discovery in grid computing system. In: Proceedings of the First IEEE/ACM International Workshop on Grid Computing. pp. 181–190. GRID '00, Springer-Verlag,

London, UK (2000), http://portal.acm.org/citation.cfm?id=645440.652826

[85] Maheswaran, M., Krauter, K.: A parameter-based approach to resource discovery in grid computing system. In: Proceedings of the First IEEE/ACM International Workshop on Grid Computing. pp. 181–190. GRID '00, Springer-Verlag, London, UK (2000), http://portal.acm.org/citation.cfm?id=645440.652826

[86] Malempati, S., Mogalla, S.: Article: Grid based approach for data confidentiality. International Journal of Computer Applications 25(9), 1–5 (July 2011), published by Foundation of Computer Science, New York, USA

[87] Mateos, C., Zunino, A., Campo, M.: m-jgrim: a novel middleware for gridifying java applications into mobile grid services. Softw. Pract. Exper. 40, 331–362 (April 2010), http://dx.doi.org/10.1002/spe.v40:4

[88] Mockapetris, P.V.: Domain names - concepts and facilities (1987)

[89] Nakai, J.: Pricing computing resources: Reading between the lines and beyond (2002)

[90] Németh, Z.N., Sunderam, V.: A formal framework for defining grid systems. In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. p. 202. CCGRID '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=872748.873260

[91] Németh, Z.N., Sunderam, V.: A formal framework for defining grid systems. In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. p. 202. CCGRID '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=872748.873260

[92] Pacini, F.: Job description language how to (December 2001), http://www.grid.org.tr/servisler/dokumanlar/DataGrid-JDL-HowTo.pdf

[93] Park, J.: Usf-pas: Study on core security technologies for ubiquitous security framework. Journal of Universal Computer Science 15(5), 1065–1080 (2009)

[94] Postel, J.: User datagram protocol. RFC 768, Internet Engineering Task Force (August 1980), http://www.rfc-editor.org/rfc/rfc768.txt

[95] Postel, J.: Internet Control Message Protocol. RFC 777 (April 1981), http://www.ietf.org/rfc/rfc777.txt, obsoleted by RFC 792

[96] R. Wolski, J. Brevik, J.S.P., Bryan, T.: Grid resource allocation and control using computational economies. In: Grid Computing: Making the Global Infrastructure a Reality. pp. 747–772. John Wiley and Sons (2003)

[97] Raman, R., Livny, M., Solomon, M.: Matchmaking: Distributed resource management for high throughput computing. High-Performance Distributed Computing, International Symposium on 0, 140 (1998)

[98] Rosado, D.G., Fernandez-Medina, E., Lopez, J.: Reusable security use cases for mobile grid environments. In: Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems. pp. 1–8. IWSESS '09, IEEE Computer Society, Washington, DC, USA (2009), http://dx.doi.org/10.1109/IWSESS.2009.5068452

[99] Rosado, D.G., Fernández-Medina, E., López, J.: Applying a uml extension to build use cases diagrams in a secure mobile grid application. In: Proceedings of the ER 2009 Workshops (CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS) on Advances in Conceptual Modeling - Challenging Perspectives. pp. 126–136. ER '09, Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04947-7_16

[100] Rosado, D.G., Fernández-Medina, E., López, J., Piattini, M.: Analysis of secure mobile grid systems: A systematic approach. Inf. Softw. Technol. 52, 517–536 (May 2010), http://dx.doi.org/10.1016/j.infsof.2010.01.002

[101] Rosado, D., Fernandez-Medina, E., Lopez, J., Piattini, M.: Developing a secure mobile grid system through a uml extension. Journal of Universal Computer Science 16(17), 2333–2352 (2010)

[102] S. Tuecke, V.Welch, D.E.L.P.M.T.: Rfc 3820: internet x.509 public key infrastructure proxy certificate profile (2004)

[103] Sirbu, M.A., Chuang, J.C.I.: Distributed authentication in kerberos using public key cryptography. Network and Distributed System Security, Symposium on 0, 134 (1997)

[104] Stamos, J.W., Gifford, D.K.: Remote evaluation (1990)

[105] Stone, G.N. Lundy, B.X.G.: Network policy languages: a survey and a new approach. In: IEEE Network. p. 10. IEEE Communications Society, US Dept. of Defence, Fort Meade, MD (2001)

[106] Subramani, V., Kettimuthu, R., Srinivasan, S., Sadayappan, P.: Distributed job scheduling on computational grids using multiple simultaneous requests. In: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. p. 359. HPDC '02, IEEE Computer Society, Washington, DC, USA (2002), http://portal.acm.org/citation.cfm?id=822086.823345

[107] Swenson, Keith (San Jose, C.: Simple workflow access protocol (June 2003), http://www.freepatentsonline.com/6574675.html

[108] T. Walsh, P.N., Dobson, S.: Review of mobility systems. In: TCD Computer Science Technical Report (2000)

[109] Takefusa, A., Matsuoka, S., Casanova, H., Berman, F.: A study of deadline scheduling for client-server systems on the computational grid. In: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing. p. 406. IEEE Computer Society, Washington, DC, USA (2001), http://portal.acm.org/citation.cfm?id=874077.876538

[110] Tariq Alwada'n, Hamza Aldabbas, H.J.T.K., Aldabbas, O.: Dynamic policy management in mobile grid environments. International Journal of Computer Networks and Communications (IJCNC) 4(2) (2012)

[111] Thompson, M.R., Essiari, A., Mudumbai, S.: Certificate-based authorization policy in a pki environment. ACM Trans. Inf. Syst. Secur. 6, 566–588 (November 2003), http://doi.acm.org/10.1145/950191.950196

[112] Verma, D.C., Sahu, S., Calo, S.B., Beigi, M., Chang, I.: A policy service for grid computing. In: Proceedings of the Third International Workshop on Grid Computing. pp. 243–255. GRID '02, Springer-Verlag, London, UK (2002), http://portal.acm.org/citation.cfm?id=645442.652675

[113] Verma, D.C., Sahu, S., Calo, S.B., Beigi, M., Chang, I.: A policy service for grid computing. In: Proceedings of the Third International Workshop on Grid Computing. pp. 243–255. GRID '02, Springer-Verlag, London, UK (2002), http://portal.acm.org/citation.cfm?id=645442.652675

[114] W. Allcock, J. Bester, J.B.A.C.L.L., Tuecke, S.: Gridftp: Protocol extensions to ftp for the grid (April 2003), http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf

[115] Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., Tuecke, S.: Security for grid services. In: High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on. pp. 48 – 57 (2003)

[116] Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., Tuecke, S.: Security for grid ser-

vices. In: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing. p. 48. HPDC '03, IEEE Computer Society, Washington, DC, USA (2003), http://portal.acm.org/citation.cfm?id=822087.823401

[117] Wong, S.W., Ng, K.W.: Performance evaluation of mobile grid services. In: Proceedings of the 2nd KES International conference on Agent and multi-agent systems: technologies and applications. pp. 557–566. KES-AMSTA'08, Springer-Verlag, Berlin, Heidelberg (2008), http://portal.acm.org/citation.cfm?id=1787839.1787903

[118] Wu. Jin, Leangsuksun. Chokchai Box, R.V., Hong, O.: Policy-based access control framework for grid computing. In: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid. pp. 391–394. CCGRID '06, IEEE Computer Society, Washington, DC, USA (2006), http://dx.doi.org/10.1109/CCGRID.2006.80

[119] Yang, K., Galis, A., Todd, C.: Policy-based active grid management architecture. In: Networks, 2002. ICON 2002. 10th IEEE International Conference on. pp. 243 – 248 (2002)

[120] Yu, C.M., Kam-Wing: Dynamic policy management framework for partial policy information. In: Advances in Grid Computing - EGC 2005 European Grid Conference - Volume 1 / 1973 - Volume 6473 / 2011. Springer Berlin / Heidelberg, Amsterdam, The Netherlands (2005)

[121] Yu, C.M., Ng, K.W.: A heterogeneous authorization policy management mechanism for grid environments. In: International Conference on Multimedia and Ubiquitous Engineering, 2007. pp. 381 –386 (2007)

[122] Yu, C.M., Ng, K.W.: Dpmf: A policy management framework for heterogeneous authorization systems in grid environments. Multiagent Grid Syst. 5, 235–263 (April 2009)

[123] Zhuk, S., Chernykh, A., Avetisyan, A., Gaissaryan, S., Grushin, D., Kuzjurin, N., Pospelov, A., Shokurov, A.: Comparison of scheduling heuristics for grid resource broker. In: Proceedings of the Fifth Mexican International Conference in Computer Science. pp. 388–392. IEEE Computer Society, Washington, DC, USA (2004)