CSEGRID PORTAL: A SECURE WEB-BASED SOLUTION FOR PROVIDING UBIQUITIOUS ACCESS TO GRID SERVICES (DESIGN, DEVELOPMENT AND IMPLEMENTATION OF A PROTOTYPE)

by

SRIKANT D. RAO

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2005

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my thesis advisor Mr. David Levine for the constant support, encouragement and guidance he has provided me with throughout my study. He has been instrumental in introducing the concept of grid computing to students like me in the Computer Science & Engineering Department at UT Arlington. I immensely benefited from the discussions that I had with Mr. Levine which has helped me shape my research under his watchful guidance.

I am especially grateful to Dr. Jaehoon Yu, from the Department of Physics at UT Arlington, for sharing his experiences and ideas on using grid technology for performing high performance computing in particle physics research. I was able to gather useful insight into the grid effort at UT Arlington and other participating universities by attending one of the Regional Analysis Center Workshop last summer.

I am also grateful to Patrick McGuigan and Mark Sosebee for their guidance and timely assistance in my initial effort of setting up a grid infrastructure. This exercise provided me immeasurable help at a later stage in setting up the grid testbed for my research work.

Furthermore, I would also like to thank my friends Nirmal Ranganathan and Vivek Somareddy, and my roommates Sudarshan Garla and Vinay Donthi for supporting me through good and bad times.

August 12, 2004

ABSTRACT

CSEGRID PORTAL: A SECURE WEB-BASED SOLUTION FOR PROVIDING UBIQUITIOUS ACCESS TO GRID SERVICES (DESIGN, DEVELOPMENT AND IMPLEMENTATION OF A PROTOTYPE)

Publication No. _____

Srikant D. Rao, M.S.

The University of Texas at Arlington, 2005

Supervising Professor: David Levine

Grid computing is emerging as a revolutionary concept for performing high performance computations on the grid. The grid infrastructure comprises of a large set of distributed computing resources that are shared across geographical and organizational boundaries. There is a need within the scientific community for friendly interfaces that can be used to obtain ubiquitous access to the grid resources. In this work, we address these needs by designing and implementing a web-based grid portal system (CSEGrid) that provides the user with transparent access to grid services like job submission, job status and resource monitoring, data transfer between grid resources,

etc. through a set of portal interfaces. The CSEGrid portal system utilizes commodity grid software and other grid middleware tools to enable users to perform grid operations through the portal interfaces.

We also describe a credential management service in our portal design to handle grid credentials that are used to authenticate the user in the grid environment. The portal system relies on this service to provide a secure and convenient access to the grid services for the user, and also prevent some of the problems that arise with mismanagement of credentials by the users. Finally, we evaluate our portal implementation against some of the other currently available grid portal projects. The implementation of the CSEGrid portal will serve as an initial prototype for providing web-based access to grid services available on some of the resource clusters at UT Arlington.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER 1

INTRODUCTION

1.1 Evolution of the Grid

Advances in Information Technology (I.T.) over the last few decades have made a tremendous impact on today's world. I.T. now forms the backbone of a diverse set of communities like banking, medicine, transportation, defense, scientific, etc. which rely on it for their computing needs in solving complex problems. The growth of the Internet and evolution of the distributed computing paradigm has seen significant technological improvements in various problem domains. It did not take long for the distributed computing concept to start gaining acceptance within the various computing communities. Projects like distributed.net [DIST.NET] and SETI@home [SETI] are considered to be some of the most successful and popular projects undertaken in distributed computing history. These projects have helped in bringing together participants that are geographically dispersed. Driven by the results achieved in these projects, there was a general consensus that distributed computing could reduce the computational time of projects. The scientific community has been particularly fascinated by the results and together with the high performance computing community proposed the creation of the "grid" [BLUEPRINT], an aggregate of infrastructure spanning multiple domains designed to enable the coordinated use of distributed high end resources for scientific problem solving. The infrastructure that constitutes a grid

comprises of a variety of geographically distributed resources such as PCs, workstations/clusters, storage systems, data sources, specialized scientific instruments, etc. The grid computing paradigm thus provides large mission oriented communities, referred to as Virtual Organizations (VO), easy access to large geographically distributed computing and data management resources [BLUEPRINT]. This collaboration technique suits the needs of scientists and researchers who are now able to work together on complex research problems despite their geographic and organizational boundaries.

Interest in grid computing has been growing tremendously with more and more institutions participating in grid-related research. The grid paradigm is still in the infancy stage and efforts are in progress to standardize the architecture, protocols, and middleware applications that constitute the grid. The Globus Alliance project [GLOBUSPAGE] with participants from various national laboratories and universities is focused on performing pioneering research in this area. As part of the Alliance project, the team developed the Globus Toolkit, a set of core services for developing grid tools and applications. The University of Virginia designed Legion, a middleware software that consists of a set of libraries, source code and executable binary files [LEGIONPAGE]. Grid software like Globus and Legion has gained a lot of popularity and is currently being used at several national laboratories as well as university research centers.

1.2 Need for Grid Portals

Using grid-enabling software like Globus, the user is provided seamless access to distributed computing resources. This means that the user is now able to access different

resources belonging to various organizations in a consistent manner even though the resources may be governed by different policies. The user is presented with the notion of the grid being one large computational system. This is truly remarkable as the grid has been able to bridge institutional boundaries and provide an environment that allows the researchers in the scientific community to collaborate on various research projects.

Although the grid software proves to be the glue to put together a grid infrastructure, using this grid middleware software still remains to be a challenge to a novice user. Installation and configuration of a grid service proves to be a time-consuming task for the scientific user who wishes to devote his time and effort in analyzing and solving complex scientific problems. In the given scenario, a researcher is required to read installation manuals, several configuration files before installing and configuring his machine to perform the desired functions. Thus, there seems to be a necessity to find a way to hide the complexities of the software from the researcher who wishes to use the services provided by this middleware software.

Grid portals are essentially a web based system offering a broad range of grid related services and access to various grid resources and are emerging as convenient mechanisms for providing the scientific community with simplified interfaces to the grid. These interfaces are designed to provide dependable, consistent, pervasive access to distributed resources. There exist a number of grid portals that are primarily driven by these motivations, like the Alliance Portal [ALLIANCE], the National Partnership for Advanced Computational Infrastructure (NPACI) HotPage [HOTPAGE], the Network for Earthquake Engineering Simulation (NEES) grid Portal [NEESGRID] and others

under construction like the GridSphere Project in Germany [GRIDSPHERE] and the NSF Middleware Initiative (NMI) Open Grid Computing Environment (OGCE) project in the United States [NSF-NMI]. Most of the grid portal projects are custom-built solutions to the challenges of connecting to disparate resources, although many of these systems are converging towards some common approach.

Some of the complexities encountered in developing a web-based interface to the grid include dealing with limitations of the client browser and the transfer protocols like the Hyper Text Transfer Protocol (HTTP), choosing suitable web server software and mechanisms to be used by the web server to interface with the grid services. Apart from these design considerations, secure access and operation of the portal is critical for the scientific users relying on the portal for grid access and also to the resource providers of the grid resources. Providing and maintaining an acceptable level of security for the portal infrastructure has a direct bearing on the successful operation of the service. The focus of this research is to investigate the security requirements of a grid portal and design such a system that would meet those requirements in an effort to provide secure web based access to the grid services.

CHAPTER 2

RELATED WORK

An increasing number of scientific projects utilizing grid based services have shown interest in development and deployment of grid portals and other application specific portals. Portals like the Astrophysics Simulation Collaboratory (ASC) Science Portal [ASCPORTAL], the WebCom computational Grid Portal [WEBCOM] , the Grid Enabled web eNvironment for site Independent User job Submission (GENIUS) Grid Portal [GENIUS], the Legion Portal [LEGION], the G-monitor portal service designed as part of the Gridbus project [GRIDBUS] have been developed portals and are presently being used by scientific research groups at various national labs and research centers. Each of these portals has been designed keeping in mind the requirements of the researchers that would like to utilize the grid resources to run experiments and analyze data.

The ASC Portal was designed to provide a visual interface using which the members of the astrophysics community could conveniently access computational grids shared by the ASC virtual organization. The architecture of the portal server is as shown in figure 2.1.

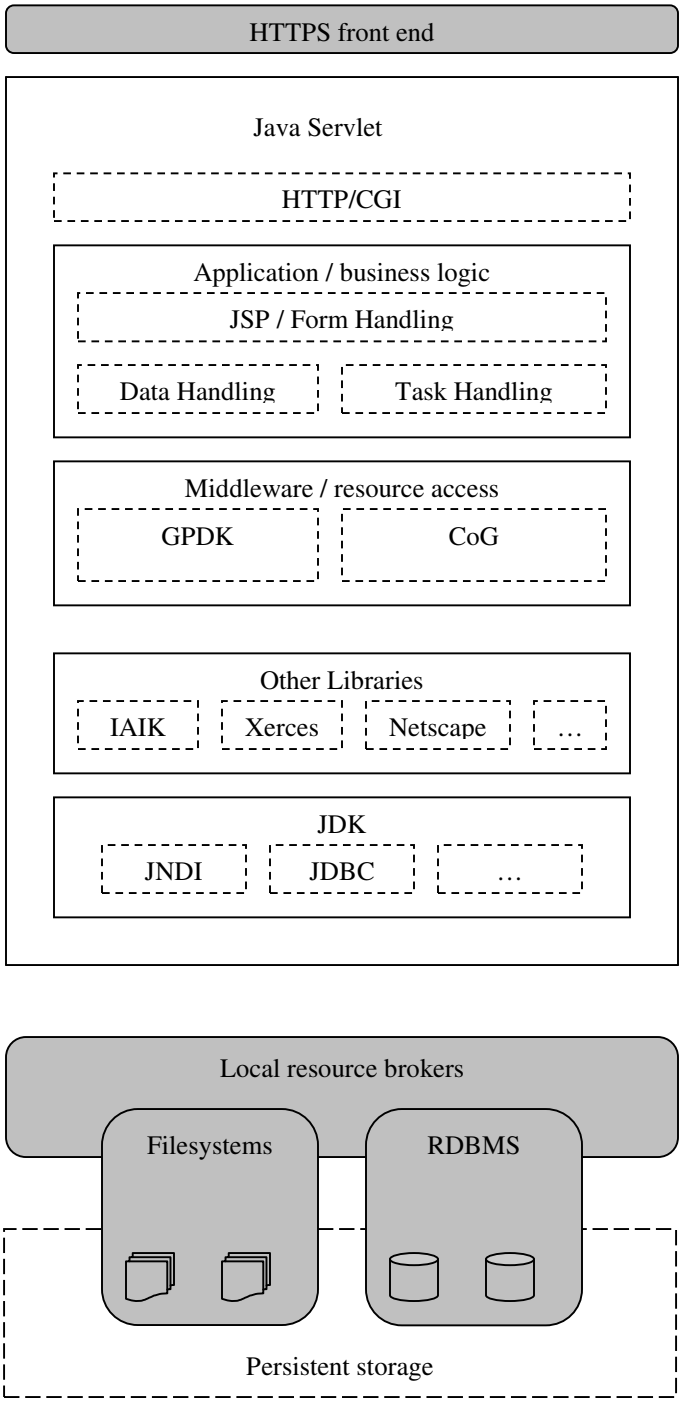

Figure 2.1 Server-side architecture of the ASC portal [ASCPortal]. The figure shows all the major software components within the server system.

The portal is powered by a web-based application server implemented using Java web services technology. Java Servlets [SERVLETS] are used to handle the functionality of the application server and Java Server Pages (JSP) [JAVA] provide a mechanism to dynamically build web pages for presenting the results and output to the user. The application server utilizes existing middleware technologies like the Java Commodity Grid (JavaCoG) [JAVACOG] toolkit and Grid Portal Development Kit (GPDK) [GPDK] to coordinate communication between the web application with lower level grid resources. Persistent storage mechanisms are utilized to manage information about users by the server. This project utilizes the MyProxy [MYPROXY] service to enable users to store and retrieve short-term proxy certificates using the web service. Additional security requirements are met using a commercial version of the Apache [APACHE] web server called Stronghold [STRONGHOLD] that comes bundled with strong cryptography.

Another portal that was developed along similar lines is the GENIUS Grid Portal. The portal was jointly developed by INFN and NICE srl as part of the INFN and DataGrid projects. The GENIUS portal is described by a three-tier architecture that comprises of the clients (web browsers), an EDG portal User Interface (UI), and the grid resources. The EDG portal UI contains various components as shown in figure 2.2.

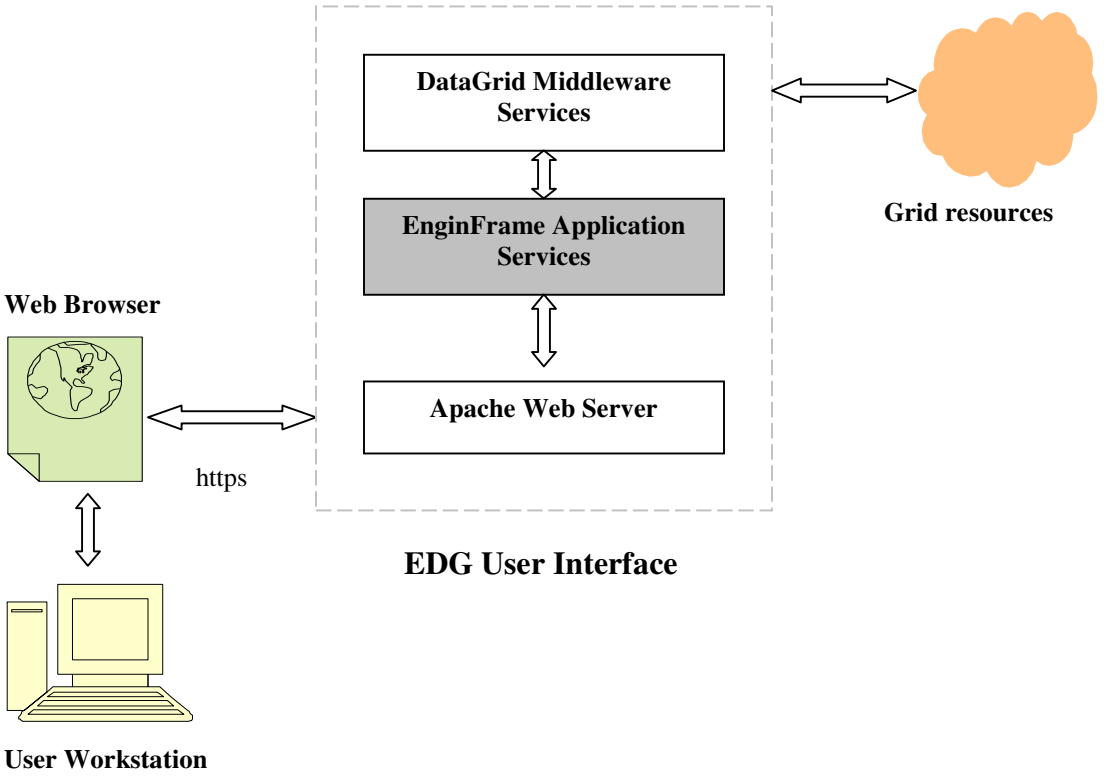

Figure 2.2 Basic architecture of the GENIUS portal [GENIUS]. The EDG user interface (UI) provides access to resources within the European Union DataGrid. Communication between the web browser and the EDG UI is secured using SSL via https. EnginFrame agents are setup on the resources on the grid that enable communication with the EDG UI. Information is transferred between the grid resources and the EDG UI in XML form which is rendered into HTML web pages and presented to the web browser.

The EnginFrame [ENGINFRAME] framework is built upon a servlet-enabled web server. The user interacts with the grid resources using the EnginFrame services. The DataGrid middleware services are used to submit user jobs and manage data on the grid [GENIUS]. The services are implemented in the form of Java Servlets which is essentially an executable code that implements the service logic that is invoked in response to a request made through a web page. The output or results of a job are presented in the form of dynamically constructedHTML pages. The GENIUS portal

provides job submission, data management and security services. Secure web communication via HTTPS, use of proxy credentials and password authentication access to the portal are some of the security measures taken by the portal developers.

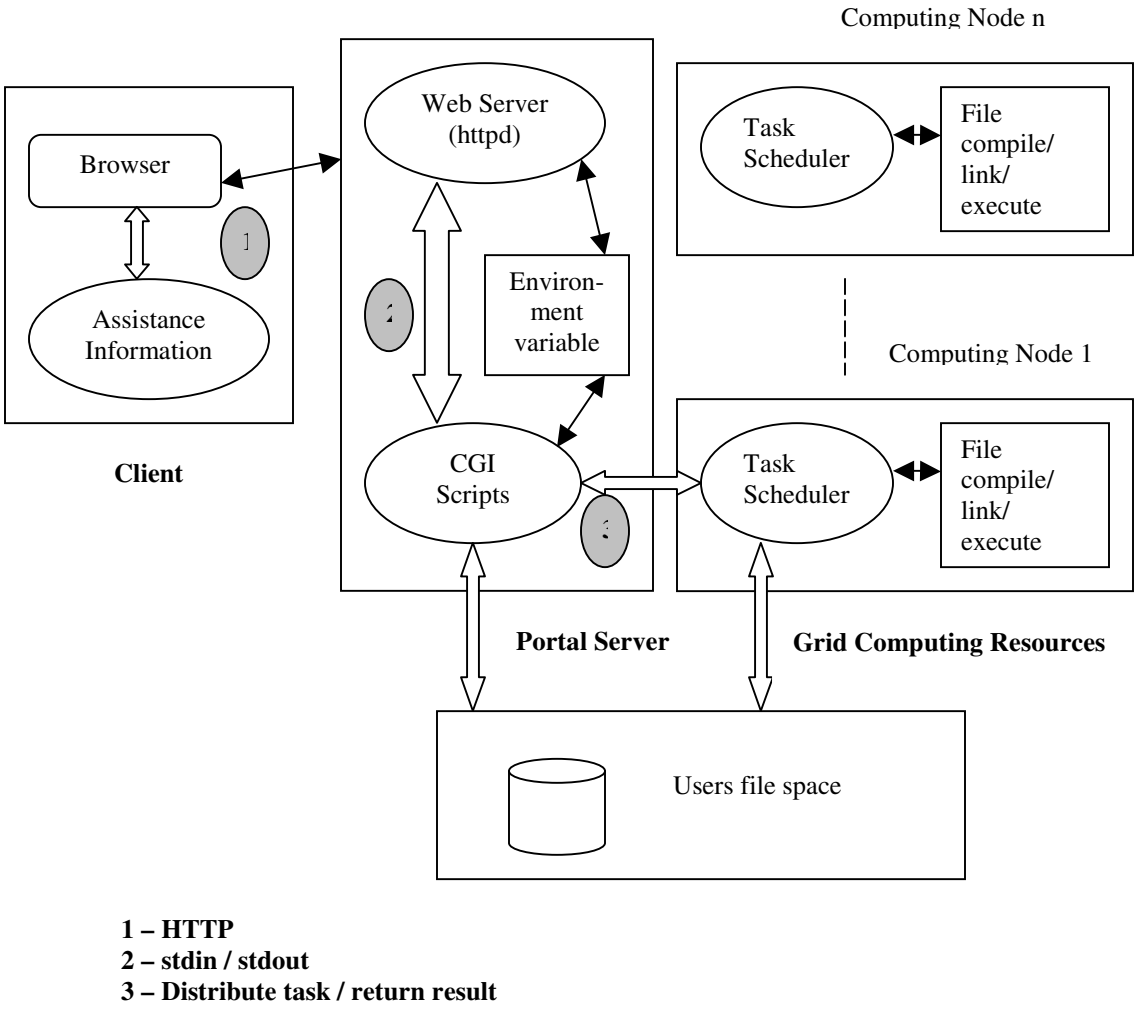

Figure 2.3 Basic architecture of the WebCom portal system [WEBCOM]. The architecture is layered into 3 tiers viz. the clients, the portal server and the computing nodes and the file storage that constitute the back end tier.

A simplified computational grid portal named WebCom has been designed and tested on a grid testbed in the Xian Jiaotong University, China as shown in figure 2.3.

The portal is used by research groups' campus wide for performing scientific computations and running simulation programs. The portal server comprises of a web server and some Common Gateway Interface (CGI) scripts that interact with the task schedulers on the computing resources to perform the requested tasks. The portal employs a password-based authentication mechanism, wherein the user is required to enter a valid username and password issued by the portal administrator to access the web pages and services. Communication between the client and the portal server is secured via Secure Socket Layer (SSL) and that between the portal server and the computing resources using Secure Shell (SSH). Access to the portal server and the resources is controlled using firewalls. The WebCom system relies of a simple password-based authentication to protect the grid services which is not suited for a large scale grid portal environment. WebCom system does not employ any strong authentication mechanisms like X.509 [RFC2511] certificate authentication which is commonly employed in the grid environment.

2.1 N-tier Architecture in Portal Development:

In the above mentioned grid portal projects, the portal developers have taken the popular n-tier based web architecture approach [ASCPORTAL] as depicted in figure 2.4. The idea here is to separate the functionality of the application into separate layers (generally 3 or more).

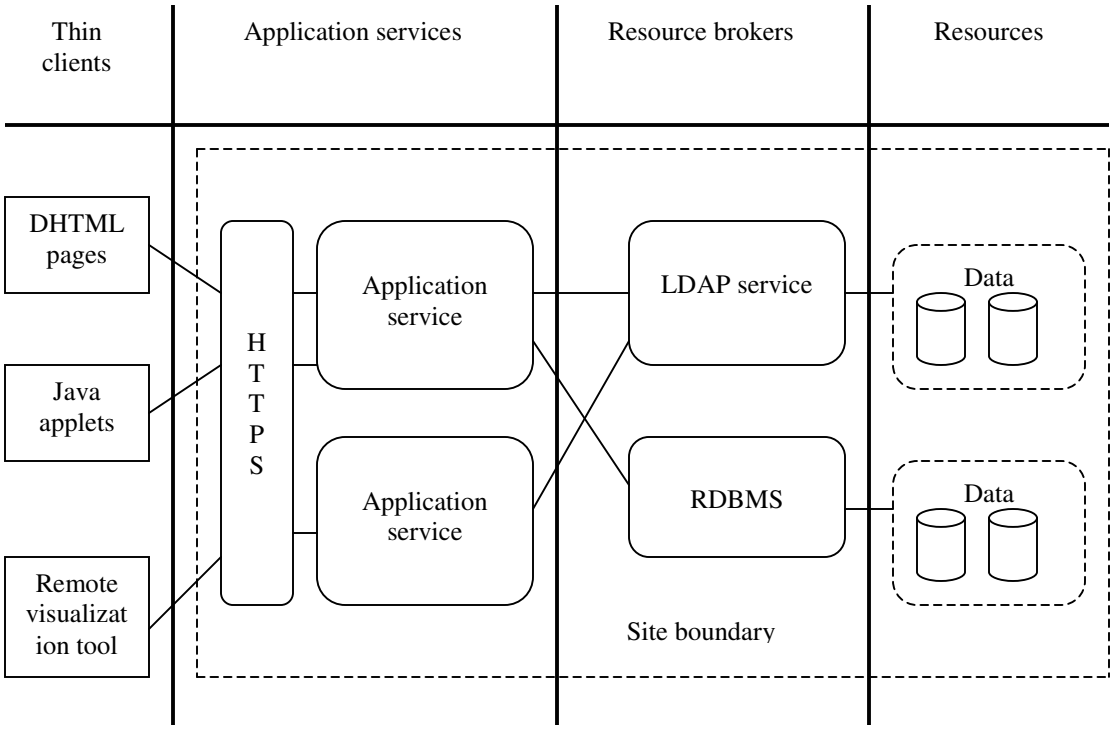

Figure 2.4 An example depicting an n-tier application. N-tier applications are designed to separate functionality into different layers. This layered approach enables resources and services to be optimally shared between the clients. The client applications are based on thin-client technology where the application logic is implemented on the server side. [ASCPORTAL]

The first layer is made up of clients that are essentially web enabled devices such as a personal computer or workstation equipped with a web browser, personal data assistants (PDAs) with web browsing capability, etc. The portal server and the application services mediated by it form the second layer. The portal server interacts with the client and obtains the user's requests which are then forwarded to the Grid resources that form the backend layer. The portal server may utilize services of a resource broker in order to obtain information about the resources in the back end layer. The resource layer

comprises of compute resources, storage devices, resource and job schedulers, and any specialized scientific instruments that may be used during computation.

2.2 Security Considerations in a PortalE nvironment:

In terms of security level offered, portals are considered to be less secure than the conventional methods employed to access data and resources at a remote site. This is because portals where primarily designed to provide the user with access to grid related services through a simple web browser. Thus the user can connect to the portal from anywhere over the internet and interact with the grid resources. Communication over the internet is susceptible to all kinds of vulnerabilities and potential attacks. To this end, portal developers have always relied on encrypting communication between the client browser and the web server, setting up some form of authentication to verify the identity of the user, and also tightening the security on the individual systems and the web server. Moreover, the grid computing environment employs Public Key Infrastructure (PKI) based certificate authentication (like Grid Security Infrastructure (GSI) which is available through the Globus software) for verifying client and host identities. These certificates are obtained from a Certificate Authority (CA) trusted by the service provider. In the existing projects, users are required to manage their own certificates; although some certificate delegation mechanisms like MyProxy [MYPROXY] has been incorporated by the portal developers while building the portals which allows the portal to store a short-term credential of the user on the MyProxy server. Although this partially solves the problem of the user having to carry his credentials to every machine from which he wishes to use the grid services, it does not address other problems like availability of

user's long term credential in the event of expiration of the short-term credential stored on the proxy server and most commonly encountered problems like credential mismanagement by the users.

Issues related to security in the grid portal environment and credential management is considered an open-ended problem and is worth investigating. The following section describes the goal of our work and highlights some of the problems that are addressed in this work.

CHAPTER 3

THESIS GOALS

The work presented in this paper is focused on developing a secure architecture for providing web-based access to services and resources in a grid environment. The system should provide a secure environment to the users who will be utilizing the services that are presented by the grid resources within the system. Interfaces to the system should be simple and easy-to-use considering the fact that the users come from a variety of backgrounds with varying levels of computer expertise. Apart from designing a secure and convenient interface to the grid resources and services, our goal is to provide a credential management service wherein the user's long-term credentials will be stored and maintained by a centralized credential management service. In summary, this research project involves design, development, and implementation of a prototype of the CSEGrid portal system architecture that will provide secure web-based access to grid services running on various organizational resources.

The following section will outline the system architecture and some of the design guidelines. This will be followed by a detailed description of specific components within the system.

CHAPTER 4

ARCHITECTURE AND DESIGN

This section is divided into a discussion of the various requirements and design issues that are need to be considered while defining the functionality of the system, followed by a description of the architectural model of the CSEGrid portal, and then a design summary describing the various components that make up the CSEGrid portal system.

4.1 Architectural Requirements & Design Issues

Grid portals have primarily been custom-built solutions to the problems encountered by people in accessing distributed resources and services that are part of a Virtual Organization (VO). Described below are some of the key design goals that were identified and have influenced the design of the CSEGrid portal system.

- Universal Access over the Internet: The system should be capable of allowing the user to be able to access resources and services provided by the portal system from any available computer (e.g. kiosks at an air terminal, office computers, or home PCs). Due to the popularity and availability of web browsers on most computers connected to the internet, a web-based approach seems to be a suitable solution.

- Simple and easy-to-use Interface: This is one of the primary goals that drive any portal development project. The interface presented to the user must be easy to understand and use without requiring any specialized training or reading of additional documentation. The user of the portal interface must not be required to download, install or configure any special tools or applications.
- Use existing grid technologies and standards: The system should be designed to work with existing grid and internet technologies and must not require the user to learn any new technology to perform the same set of tasks that were carried out using non web-based access.
- Security: The system should support use of secure communication and encryption technologies at all levels in the design in order to protect the integrity and confidentiality of the information provided by the user to the system. Suitable authentication, authorization and access control schemes must be identified in order to secure the system.
- Credential Management: The system must provide a secure credential management facility to create, renew and store the user's grid credentials. The credential management policies and practices must be documented and made available to the users.
- Performance and reliability: The portal system provides a front-end interface made visible to the user. This makes the interface a single point of entry to the system. The system should make provision for adding sufficient redundancy into

the system by replicating critical information and key entities to make the system tolerant to failures.

4.2 Architecture

The architecture of the CSEGrid portal system is shown in figure 4.1.

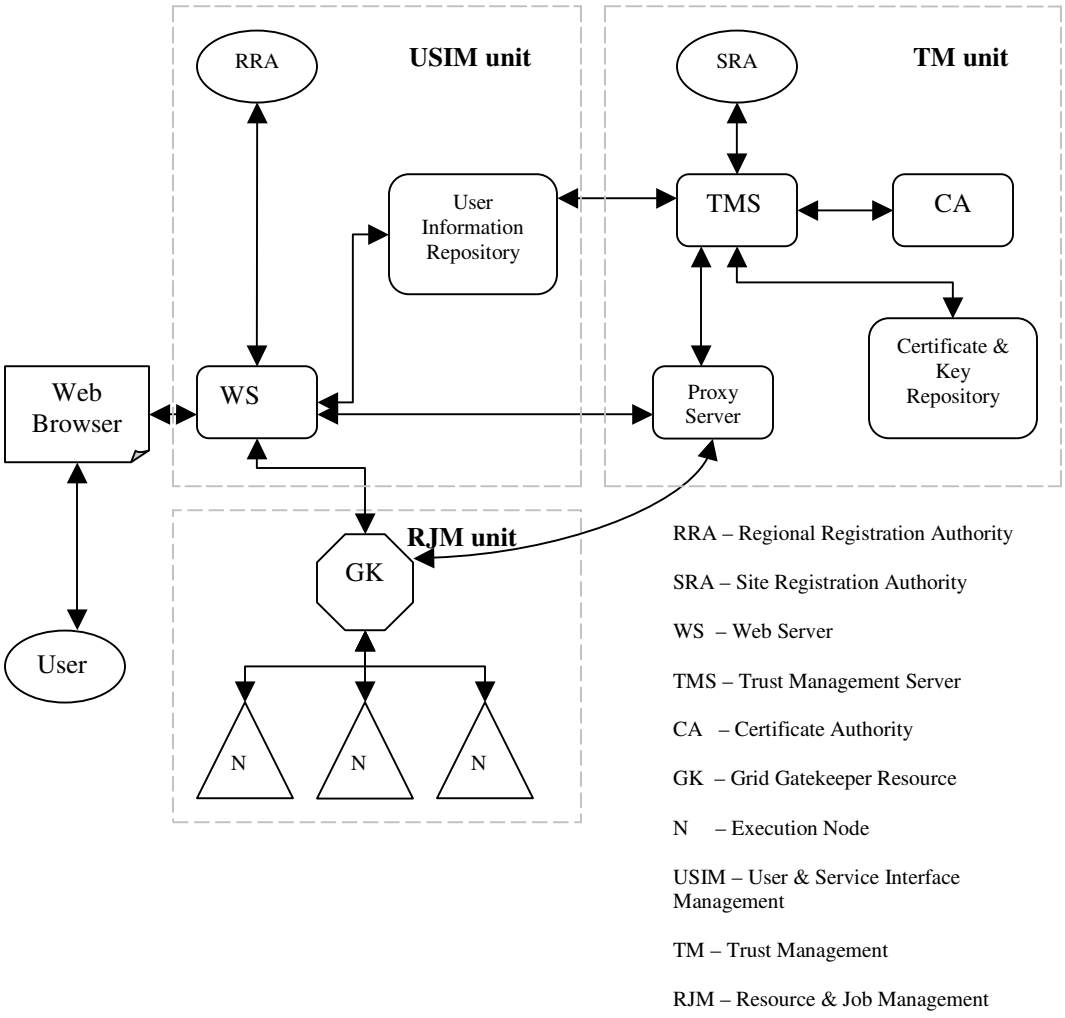

Figure 4.1 Basic architecture of the CSEGrid portal system. The architecture is partitioned into three categories viz. the USIM unit, TM unit and RJM unit based on the functionality provided by the portal system. The double-headed arrow between two entities shown in the diagram indicates an interaction between the entities.

It outlines the various entities within the proposed framework and the interactions that occur between them. The system is partitioned based on the functionality managed by the system into three major categories:

i. User and Service Interface Management unit

ii. Trust Management unit

iii. Resource and Job Management unit

Each of these categories contains one or more hardware and software components. Components that are responsible for providing a specific service are referred to as *service entities*. The functionality handled by each of the categories is described as follows:

i. User and Service Interface Management (USIM) unit: It presents the user with interfaces that may be used to interact with the system. It is also responsible for handling service requests received from the user and submitting them to the appropriate management unit.

   Communication between USIM unit and the Trust Management unit typically involves service entities that handle:

   - User's request for creation/renewal of grid credentials.
   - User's request for creation/delegation of proxy credentials for storage.
   - Retrieval of delegated proxy credentials for usage by the web server on user's behalf.

   Communication between the USIM unit and the Resource and Job Management unit involves service entities that handle:

   - Submission of user's job description.

- User's request to cancel a previously submitted job.
- User's request to display status of his submitted jobs.
- User's request to display information about advertised grid resources.

Apart from brokering service requests to the other management units, the USIM also handles user authentication, creation and management of portal accounts for the users. These functions are handled by the service entities within the USIM unit.

ii. Trust Management (TM) unit: It is responsible for providing the credential management functionality within the system. Present grid technology employs the certificate-based authentication mechanism for validating user and service identity on the grid. The certificates are encoded in the X.509 certificate format, a standard data format for certificates established by the Internet Engineering Task Force (IETF) [GLOBUSPAGE]. Hence, the TM unit is required to perform operations such as creation of certificate signing requests (CSRs) or certificate renewal requests *on behalf of the users*, and also various certificate management operations like renewal, revocation, deletion and signing of certificates in the *role of a CA*. The prototype of the proposed system would utilize the services of a self-signed CA that will issue signed certificates to various grid entities within the system and to the users whose certificate requests are approved by the Site Registration Authority (SRA). The Certificate Policy (CP) and Certificate Practice Statements (CPS) of the CA [see APPENDIX A] are to be documented and made available to the users that intend to obtain certificates from the CA.

Communication between the TM unit and the Resource and Job Management unit generally involves service entities that handle:

- Submission of request for renewal of user's temporary credentials.
- Request for creation of credentials for grid resources and services.
- Renewal of credentials issued to grid resources and services.

iii. Resource and Job Management (RJM) unit: This unit is primarily responsible for handling a user's job request and scheduling them from execution on a specified grid resource. It is assumed that the service entities within the RJM unit trust the service entity (the Certificate Authority) issuing credentials to the users. A detailed discussion of security considerations will be made in a later section.

This section provided an architectural overview of the proposed portal framework. The system was decomposed into three basic units based on the functionality that the system intends to provide. The management units are required to exchange information to collectively provide the service requested for by the user.

4.3 Design

The design phase of the CSEGrid system extends the discussion of the logically arranged managements units. The design process involves construction of management unit with various service entities, representation of the data handled by these service entities, identifying the protocols that would be utilized to perform message passing between the service entities.

4.3.1 User and Service Interface Management Unit Design

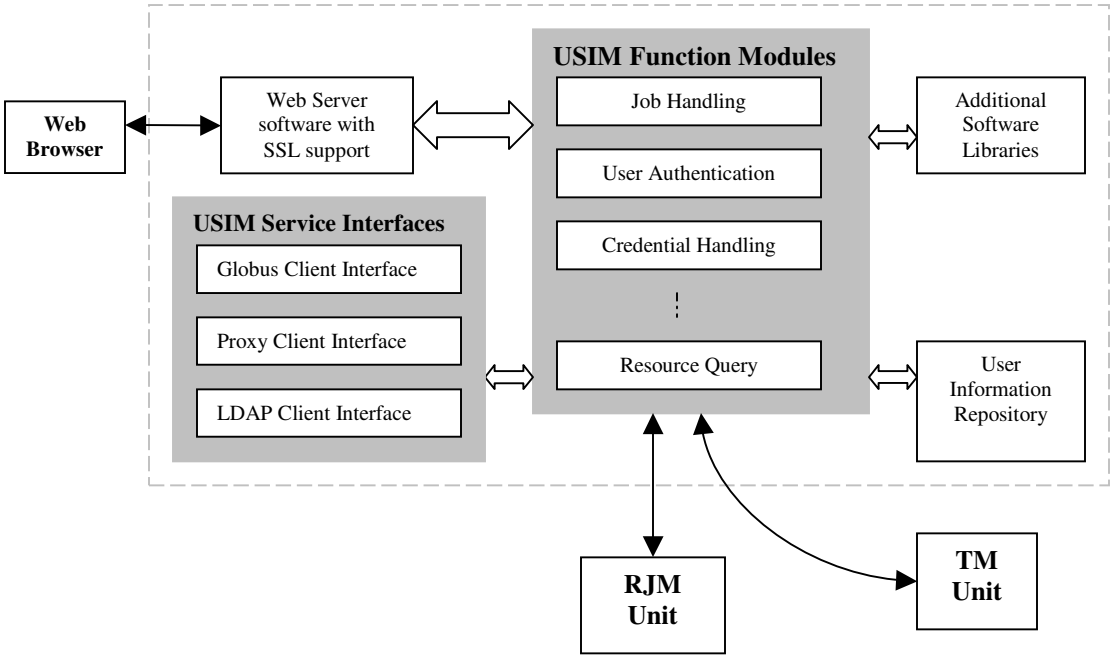

Figure 4.2 Basic design components within the USIM unit. The web server provides the users with a web-based access to the interfaces provided by the CSEGrid portal. The functionality of the USIM unit is organized into a set of modules that will implement the desired operations. The USIM function modules interact with various service entities like the information repository, the Gatekeeper service entity in the RJM unit, etc. using the client interfaces that are made available to it. The information pertaining to the users is stored in an information repository.

The USIM unit as described earlier is responsible for providing a front-end interface to the system and to handle the service requests from the user. The design of the USIM unit is as depicted in figure 4.2.

The design contains various components that are described below:

1. Web Server: Communication between the user and the portal system is set up using a web server. This allows the users to access the services provided by the portal system using any standard web browser (e.g. MS Internet Explorer,

Netscape Mozilla, etc.). The user is not required to install any additional applications or software packages on the client system. The data sent to the web browser by the web server is in Hyper Text Markup Language (HTML) format. Web browsers on the client machine are able to parse the HTML data and display it in the form of a web page on the client browser. The web server used in the construction of the USIM unit should be capable of supporting encrypted communication. This may be achieved with an encryption module that is built into the server or a software entity that sits between the browser and the web server to perform encryption. The user will be sending sensitive data to the web server (like username, password, and personal information for registration). By transmitting this data in clear text (which is the default form selected by most web servers) over the internet, the data is exposed to various threats like man-in-the middle or replay attacks and commercial sniffers that are able to steal data in transit without being detected. Using encrypted communication would make it difficult if not impossible to carry out such attacks. The standard protocol used for web communication is HTTP, although a secure alternative (HTTPS) is recommended.

2. USIM Service Interfaces/APIs: In order to allow the function modules to farm out service requests to various service entities within the USIM unit as well as to entities in other units, the USIM unit utilizes various low level client APIs like Globus [GLOBUSPAGE], MyProxy [MYPROXY] and a few other s. Most of the standard services used in the grid environment come with APIs that may be

downloaded, installed and made use of almost immediately. Whenever a new service needs to be implemented, a corresponding client API if available may be added to this component to provide necessary functionality. These service APIs are installed on the machine that runs as the web server.

3. Function Modules: The front-end provided by the web server is the only visible portion of the system to the user. The actual functionality of the system is hidden behind this front-end layer in the form of various modules. For each service that is to be handled by the USIM unit, a module is designed that implements the necessary logic. All the modules reside on the web server and are invoked by the server whenever a corresponding service is requested for by the user. The input to the modules is the information supplied by the user's browser to the web server which may be a simple HTTP GET request to display a static HTML page or some HTML input parameters received as form-based input requesting for a particular service. In the latter case, the concerned module will parse the input, process the data, and then either send back an HTML formatted output back to the user or forward the data to a service entity that can handle the request. The USIM function modules reside on the machine running as the web server.
4. User Information Repository: As a measure of securing the services provided by the system, the users are required to obtain an account on the portal system. Only authenticated users are given the privilege to access the services provided by the portal. The information pertaining to users who are granted an account on the system is maintained in an information repository. The repository should support

operations such as adding a new user entry, deleting an existing entry, searching for an entry or a particular attribute within an entry. The information repository is accessed by the web server while authenticating portal users and also by the Trust Management Server (TMS) in order to retrieve user information to create Certificate Signing Requests (CSRs). The storage unit designed to function as the user information repository is chosen to be an LDAP Directory Server. Some of the advantages of using an LDAP directory service over a traditional database in the design are as follows:

- Information Extensibility: The directory service provides flexibility in the type of information that may be stored and the way the information may be organized and searched within the directory server. This is made possible through the use of directory schemas. New schemas may be designed if necessary which along with the set of pre-defined schemas made available by the directory vendor extend a lot more flexibility to the design than databases.
- Standards and interoperability: The LDAP directory service provides a standard protocol and API to access information stored in the directories. This allows any application (vendor based or custom built) that uses the LDAP client APIs to communicate with the LDAP directory server. Moreover, these applications can be made to work with any directory server that supports the standards. In contrast, database access and

application migration is a little bit more complicated with lack of a proper standard.

- Performance: The performance requirement for the directory server is 10 to 100 times greater than for the database in terms of the number of queries handled per second. The directory server supports simple queries as compared to a complex database transaction. This high performance demand is placed by the high read-to-write ratio on the directory. As such the directory service is ideally suited for applications that require fewer update operations than the read or search operations. This suits the requirements of our user information repository.
- Security: The LDAP security model provides a great deal of flexibility in securing the information stored in the directory. The LDAPv3 supports use of independent security frameworks like Simple Authentication Security Layer (SASL) to provide authentication services. The LDAP access control model provides mechanisms for specifying fine-grained access control policies.

The LDAP data model organizes information in the form of *entries*. Each entry contains a collection of attribute types and values. The attribute types specify data elements that characterize the object with which they are associated. Related attributes are grouped together to define an *object* class [RFC2252]. As an example, the object class definition of a *CSEGrid* user is shown in figure 4.3.

```
Objectclass ( NAME  'CSEGridUser'
              DESC  'CSEGrid portal user'
              SUP   inetOrgPerson
              MUST  ( userID
                      userPassword
                      commonName
                      surname
                      organizationName
                      telephoneNumber
                      emailAddress
                      postalAddress
                      stateName
                      countryName
                    )
              MAY   (
                      csegrid-department
                      csegrid-RA
                    )
```

Figure 4.3 An example of an object class definition describing a CSEGrid user. The CSEGrid user extends the inetOrgPerson object class. All attribute types relating to the CSEGrid user are grouped within the object definition. The 'MUST' keyword marks all attributes that are required to be present in an entry. The optional attributes are specified using the 'MAY' keyword in the class definition.

In order to reference information in the directory, the LDAP model defines a *namespace*. Entries are organized with this namespace in a hierarchical fashion to form what is called a *Directory Information Tree* (DIT). The name of an entry within the namespace is formed by one or more unique attribute values and is called the *Relative Distinguished Name* (RDN) of the entry. Each entry is referenced by its *Distinguished Name* (DN) which is constructed by traversing the DIT until the root node is reached and adding each entry along the way to the RDN of the entry. The RDN and the DN are written is little endian order as shown in the example in figure 4.4.

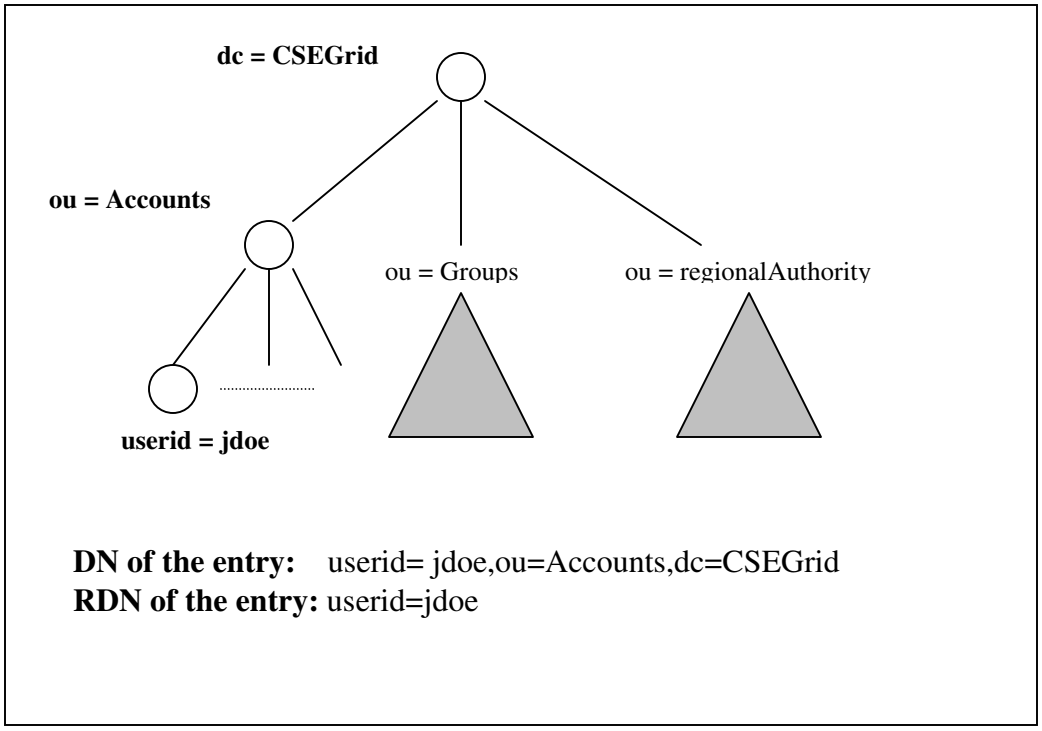

Figure 4.4 An example showing a section of the Directory Information Tree (DIT). The entries are ordered according to the information that is needed to be stored. In this case, Accounts, Groups, and regionalAuthority form the components under which child entries are stored. The child entry shown in the figure is identified with the userid attribute which is assumed to be unique to all other sibling entries. This forms the RDN of the child entry. The DN of the entry is constructed in a bottom-up fashion by traversing the DIT from the child entry until the root node is reached. The DN is written is little endian form where the least significant information appears first.

5. Additional Software Libraries: The service entities within the USIM unit may choose to use software libraries like an authentication library. These libraries might be needed to achieve desired functionality while processing a service request.

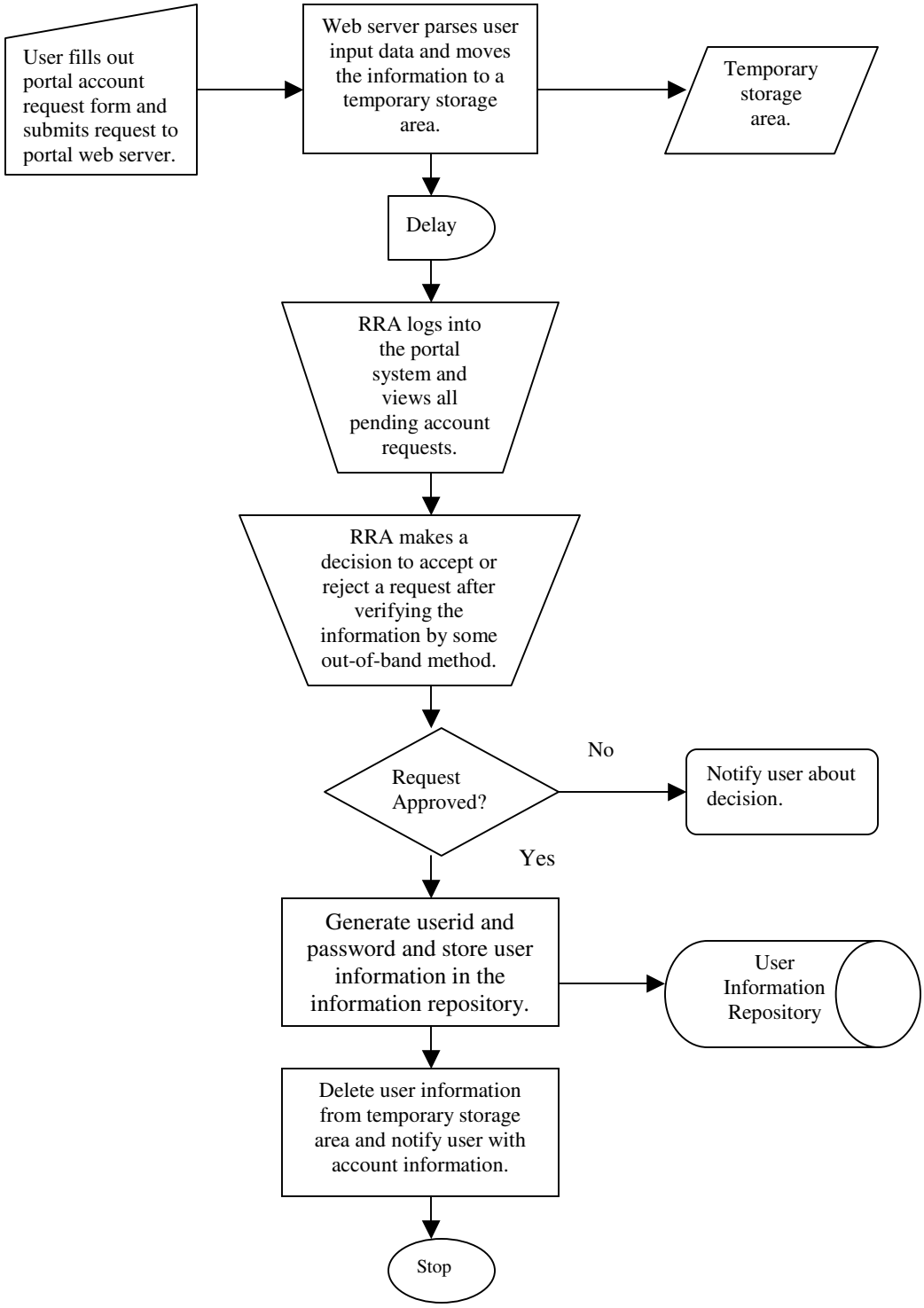

Figure 4.5 Workflow diagram illustrating the portal account creation process. The Regional Registration Authority (RRA) is a person authorized by the SRA and the portal administrator to verify the identity of the user requesting for an account and determine is account request may be approved.

4.3.2 Trust Management Unit Design:

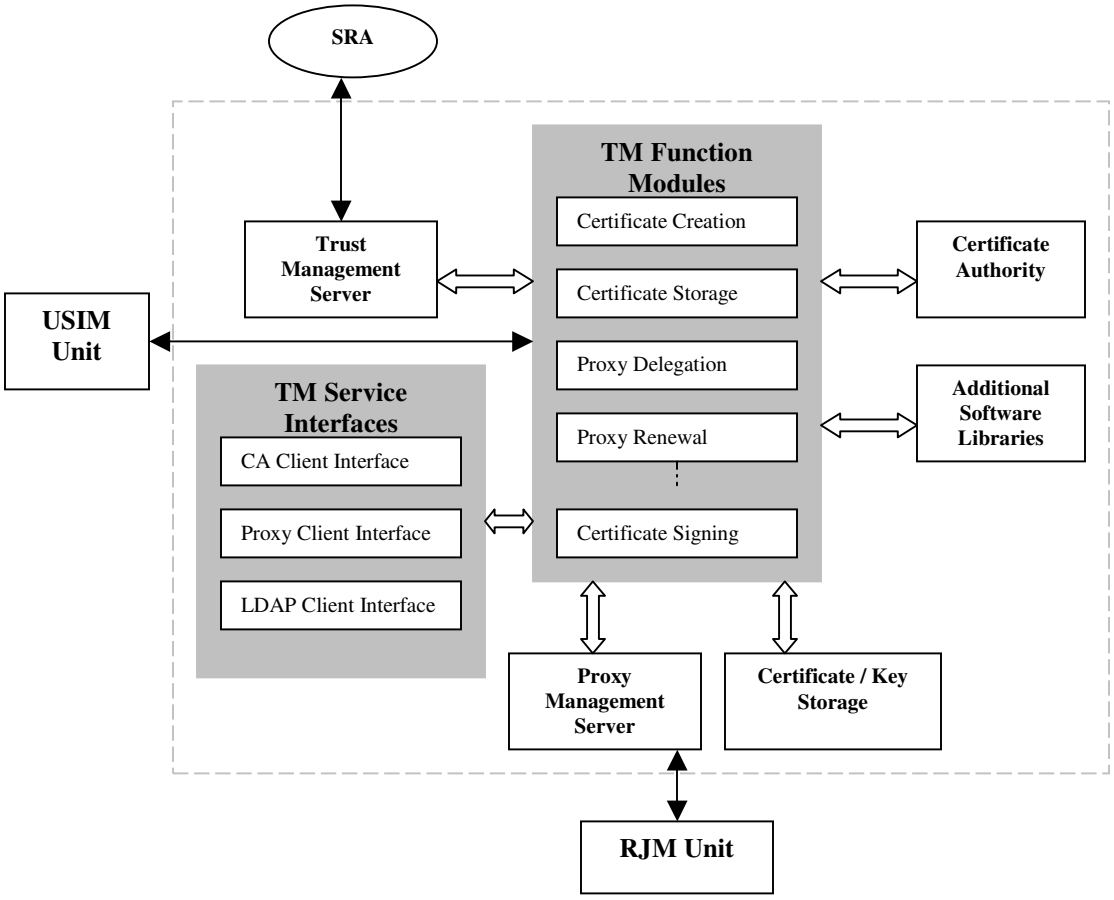

Figure 4.6 Basic design components within the TM unit. The Site Registration Authority (SRA) is a person authorized to carry out credential management operations provided by the TM unit. The Trust Management Server (TMS) provides a secure shell access to the SRA to interact with various TM components like the Certificate Authority (CA), Proxy Management Server (PMS), and the credential storage to handle the credential management request received from the USIM unit. The SRA invokes an appropriate module to carry out this task.

The components that constitute the TM unit are as depicted in figure 4.6. The functionality provided by this unit is critical to the system as the credibility of the services provided by the other units is based upon it.

The design details of the various components are described as follows:

1. Certificate Authority: The Certificate Authority (CA) for the prototype of the system being designed is a central authority that performs the following functions:
    - Issues user and host certificates that are signed using the CA's private key.
    - Renews or revokes user and host certificates that were issued earlier.
    - Maintains a list of certificates that were revoked in the form of a Certificate Revocation List (CRL).

    The CA creates a self-signed certificate that establishes the identity of the CA to the subjects. A *subject* is the identity specified as a field in the certificate request of an entity. The CA certificate is made available to the users and hosts that wish to trust the CA. All hosts within the proposed system will obtain certificates from this CA. Certificate signing requests received from the Trust Management Server is evaluated by the CA administrator before signing. Signed requests are sent to the Trust Management Server using a secure transfer protocol like secure ftp (sftp) or secure copy (scp). If any CSR is not acceptable for signing, the CA administrator notifies the Site Registration Authority (SRA) by sending an email specifying the subject of the CSR and identifying the reason for the denial.
2. Trust Management Server (TMS): The Trust Management Server is a key component of the TM unit. It is responsible for performing various credential management operations by invoking a relevant module from the TM Function Modules component. The operations performed by the TMS are initiated by the Site Registration Authority (SRA). The SRA is an authorized person with

privileges to handle the credential management service at the site. The TMS must be a highly reliable and a well secured system as it is the central entity that commands other service entities within the TM unit to perform the trust operations. The TMS provides secure shell (SSH) access to the SRA to carry out credential management operations.

3. Certificate/Key Storage: The signed user certificates and private key, which constitute the user's credential, need to be stored in a secure location to prevent compromise of the credentials. The LDAP Directory Server is chosen as the storage unit as it offers various benefits as mentioned in the discussion of the user information repository within the USIM unit. The credential information is organized in the form entries within the directory namespace. Entries are identified by the RDN that corresponds to the *userid* of the user's portal account. An example of a user credential object class, *CSEGridCert*, is shown in figure 4.7.

```
Objectclass ( NAME  'CSEGridCert'
              DESC  'CSEGrid user credentials'
              SUP   'pkiUser'
              MUST  ( userID
                      userPrivateKeyPassphrase
                      userCertificate
                      userPrivateKey
              )
             )
```

Figure 4.7 An example of an object class describing a CSEGrid user credentials. The object class is used to describe the type of entry that is stored in the Certificate and Key Storagr LDAP server.

4. Proxy Management Server (PMS): The Proxy Management Server is an integral part of the portal system. The resources in the grid environment that employ the Globus middleware technology are protected by the Grid Security Infrastructure (GSI), a set of protocols, libraries, and tools that allow users and applications to securely access grid resources [MYPROXY]. The GSI supports delegation of a short term credential created by the user using his long term credential to the grid resources. Delegation of proxy credentials to the grid resources offers the following benefits:

    - The jobs submitted by the user may now be able to run unattended on behalf of the user.
    - The user is not required to authenticate with each and every grid resource involved in processing the user's job. Thus offering the benefits of single sign-on (SSO) access to the system.
    - Since these proxy credentials are a short-term binding between the user's identity as specified in the user's long term credential and the alternate private key, compromise of the proxy credentials has a far less negative impact than a compromise of the user's long term credentials.

    In a grid portal environment, the portal web server requires that the users delegate their credentials to the portal server for it to be able to act on the user's behalf. By delegating the credentials, the server would be able to initiate and manage grid operations for the user on the grid resources. But due to the inability of the current web security protocols, delegation of credentials by the user cannot

be supported. In order to overcome this problem, a proxy server is used to store the temporary proxy credential of the user that may be retrieved by the web server at a later time. The proxy service employed by the CSEGrid portal system is the MyProxy credential repository system.

5. TM Service Interfaces: The TM unit employs a number of client tools and APIs in order to support communication with various service components. Some of the service interfaces supported are the MyProxy client tools, LDAP client tools and libraries, the Globus client tools and API.

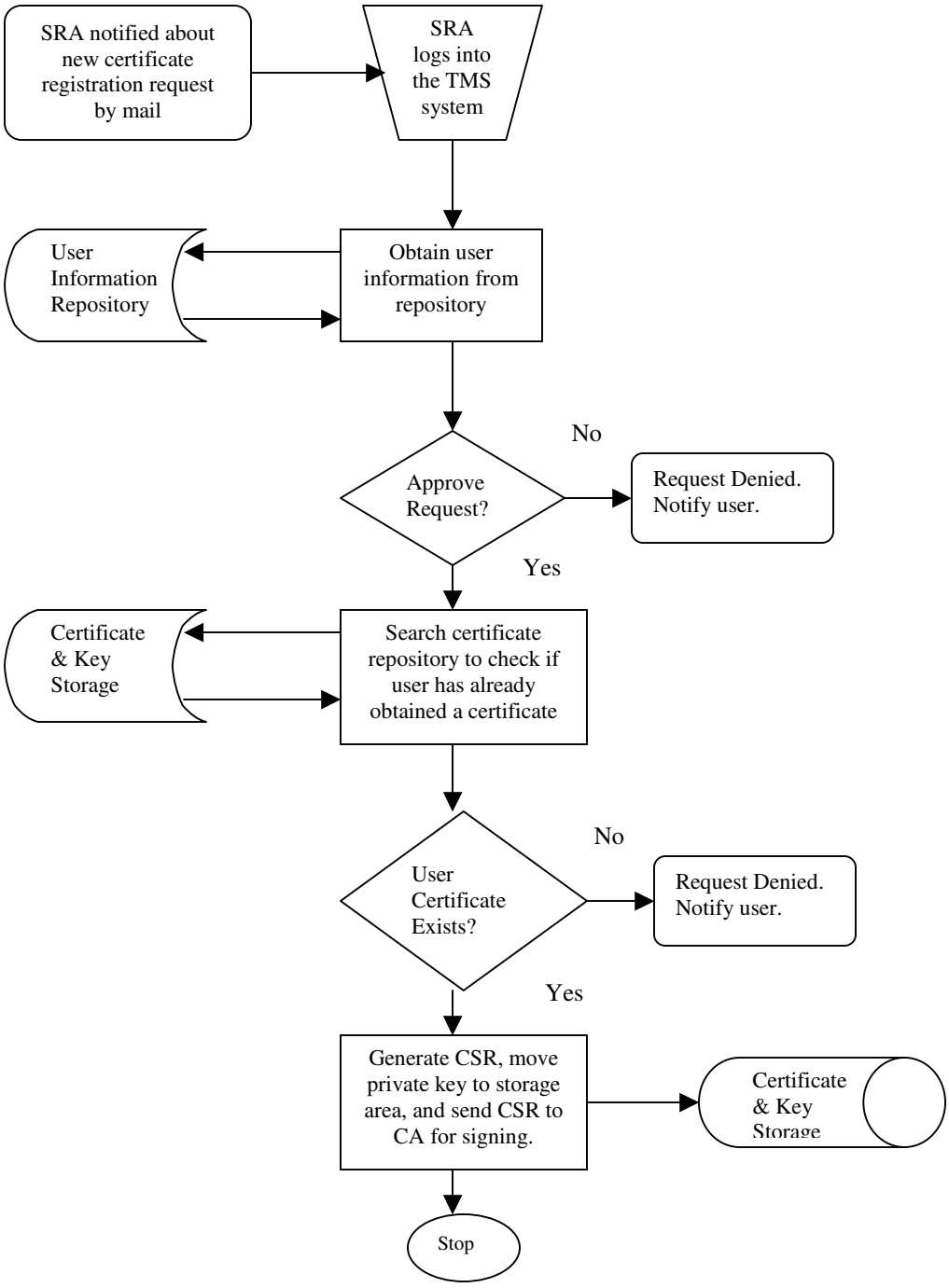

Figure 4.8 Workflow diagram indicating the X.509 certificate creation process. The process terminates after sending the Certificate Signing Request (CSR) to the CA. At a later time, the signed certificate is received by the TMS and moved to the Certificate & Key Storage.

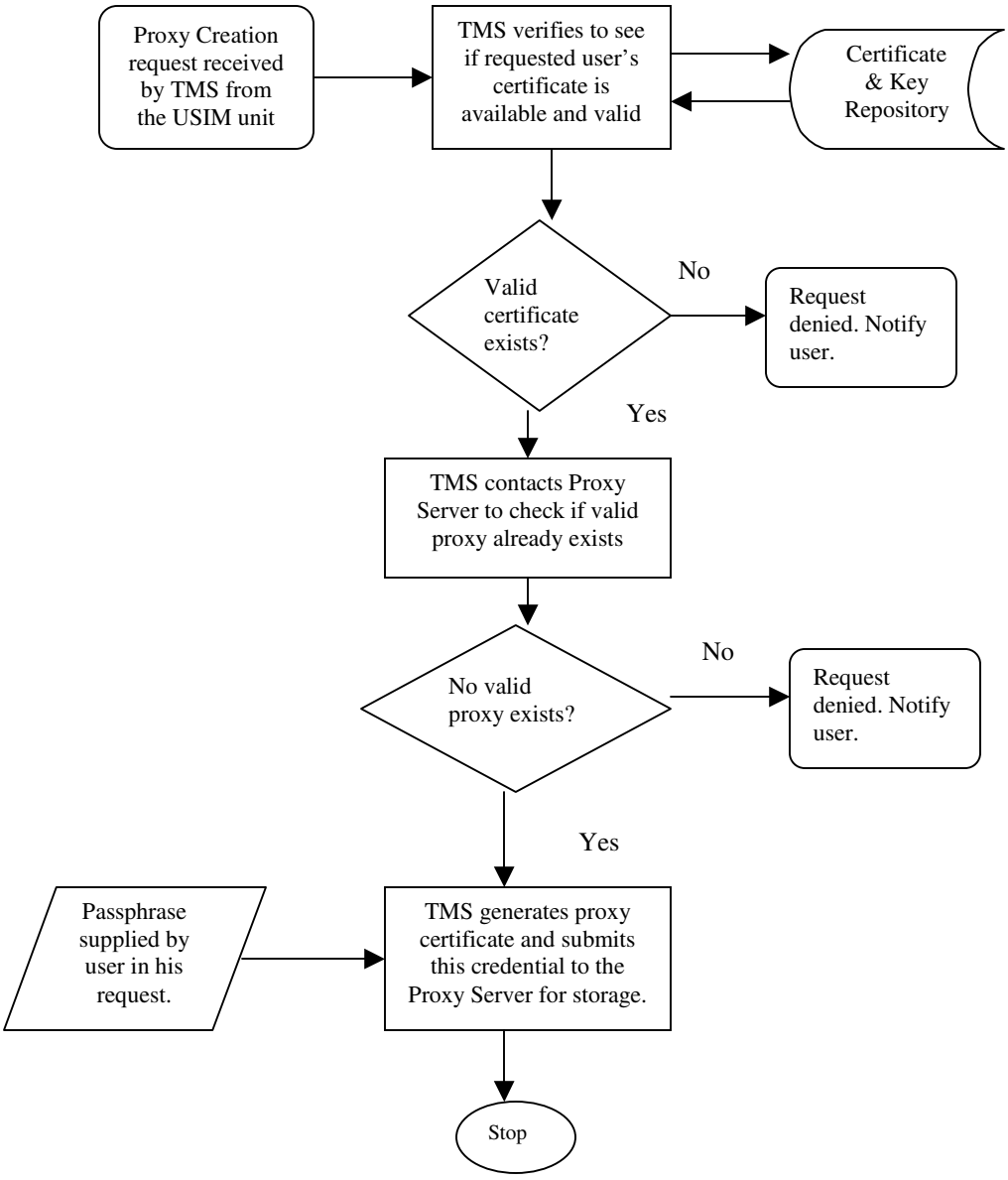

Figure 4.9 Workflow diagram describing the proxy certificate creation process. In order to delegate a proxy certificate to the proxy server, the user must have previously obtained a long term certificate from the CA and this certificate must be available to the TMS in its certificate repository. The delegated proxy is stored in the proxy server repository for future retrieval by the web server.

4.3.3 Resource and Job Management Unit Design:

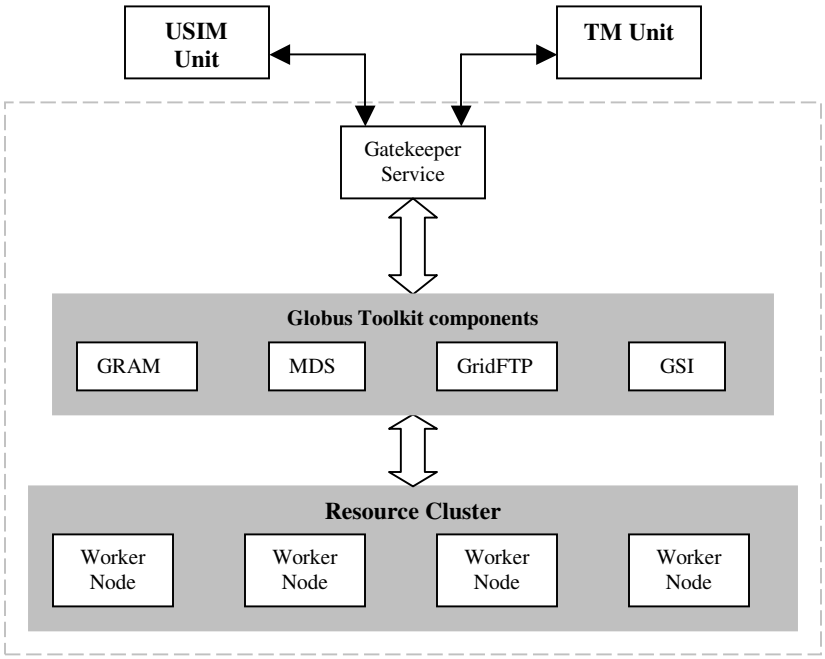

Figure 4.10 Basic design components within the RJM unit. The Gatekeeper handles requests received from the USIM unit. The Gatekeeper and the worker nodes within the resource cluster use the Globus middleware software to provide grid-level access to the resource request from the user. The solid bi-directional arrows indicate that the cluster resources and the gatekeeper utilize the core grid services provided by the Globus Toolkit components to handle the user requests.

The design components within the RJM unit are shown in figure 4.10. The RJM unit comprises of a cluster of computers that form the resources available to the user as part of the grid environment and a Gatekeeper service that handles communication with other units in the system. The Globus Toolkit provides grid services for data management and movement, resource monitoring and discovery, resource acquisition and management, and security through a set of components that implement these services. Table 4.1 describes each of these components and the services that they implement.

Table 4.1 Basic services provided by the Globus toolkit components.

Component Name	Service	Description
Globus Resource Allocation Manager (GRAM)	Resource Management	Provides global resource allocation services.
Globus Metacomputing Directory Service (MDS)	Resource Discovery & Monitoring	Provides distributed access to state and structure of resources on the grid.
Grid File Transfer Protocol (GridFTP)	Data Movement	Implements the data transfer protocol for data management between grid resources.
Grid Security Infrastructure (GSI)	Security	Provides authentication and authorization support.

1. Grid Security Infrastructure (GSI): The Globus Toolkit enables authentication and secure communication between grid entities over an open network using GSI. GSI is based on public-key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol [GLOBUSPAGE]. Some useful features of GSI are as follows:
    - Provides support for secure communication (authentication and/or confidentiality) between grid entities.
    - Provides support for single sign-on to grid users, and allow delegation of credentials in a situation where the computation spans multiple resources or sites.
2. Metacomputing Directory Service (MDS): The Metacomputing Directory Service provides the means to monitor a changing grid environment. Grid users, application executing on the grid resources and other grid services need to know the structure and state of the system in order to adapt their behavior accordingly. MDS stores information about system components (like operating system type, memory and CPU capacity, network bandwidth and latency, etc.). This

information is then published and made available using a directory service. MDS also provides a rich set of tools and client APIs for applications and users to discover and access the information stored in the directory server.

3. Globus Resource Allocation Manager (GRAM): The GRAM is the lowest level of the Globus resource management architecture [FOSTER98]. The GRAM comprises of two major software components namely the Gatekeeper and the Job Manager. The Gatekeeper upon receipt of a request from the portal server verifies the identity of the user requesting the service and determines if the user is authorized to access the service, by determining if the user's grid identity is mapped to a local account that has the privileges to access the service. The mapping rules are specified in a configuration file called the *grid-mapfile*. The user's job request is authorized based on his grid credentials and the policy specified in the *grid-mapfile*. GRAM handles these security procedures by making calls to the GSI. The Gatekeeper then creates a Job Manager Instance (JMI) [KEAHEY02] to handle the job request which executes under the user's local credential. The JMI then parses the job request and submits this request to the local scheduler. The scheduler implements the site's policy which controls the job chosen for execution, the time when the job begins execution and the resources on which it would run. Once the job begins execution, the JMI monitors the job status and handles all job management requests (e.g.: suspending a job, or querying the status of a job, etc.). The job manager terminates when the job that it

handles completes execution or is terminated by the user. The steps involved in handling a job request is illustrated in figure 4.11.

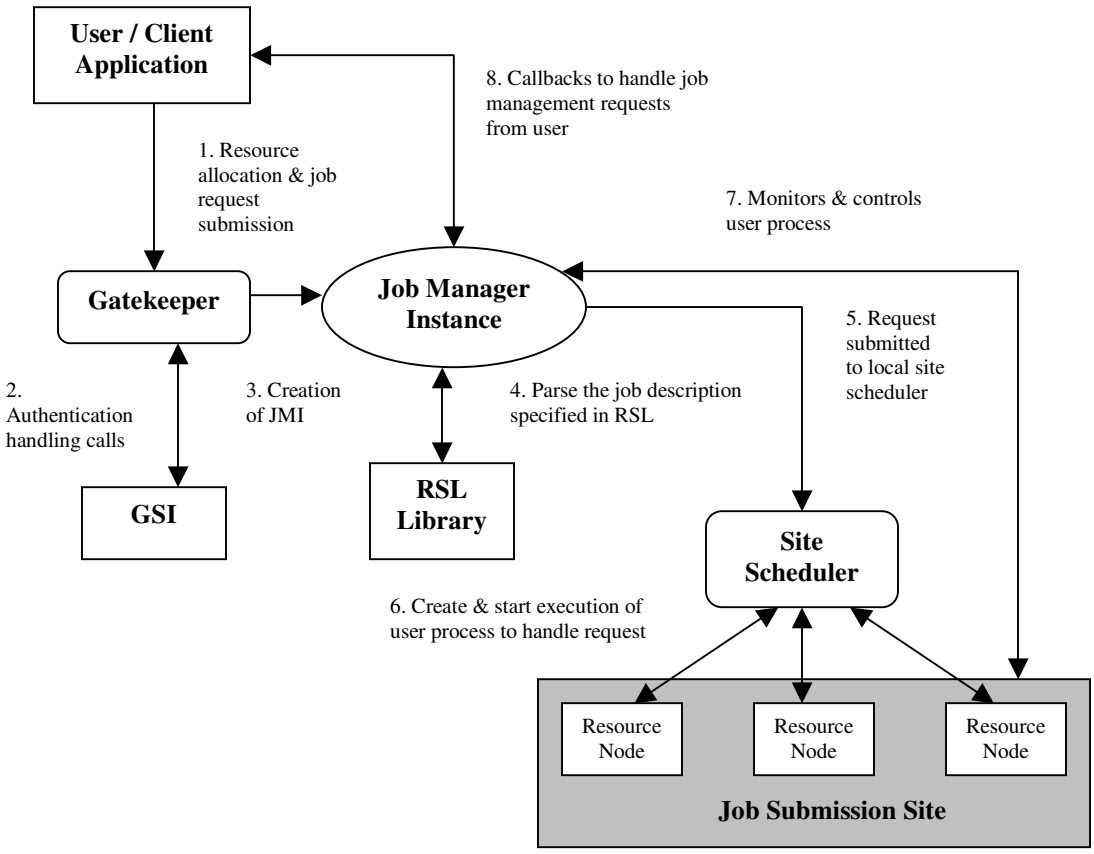

Figure 4.11 Job handling process implemented by GRAM. The Gatekeeper, the Job Manager Instance which is a software process spawned by the Gatekeeper, along with other software components constitute the GRAM. The JMI interacts with a local resource management system (like PBS, LSF, etc.) and submits the client request. The scheduler available at the site is responsible for selecting suitable resources and scheduling the job for execution.

4.4 Security Considerations in the Design

In the construction of the various management units, security forms a fundamental issue that needs to be addressed in order to provide secure and reliable services to the user. Designing for security within the management units requires an

understanding of what needs to be protected, identifying the vulnerabilities within the system (which comprises of management units) and the threats that it is exposed to, and finally deciding on how to secure the system by identifying suitable mechanisms.

4.4.1 Understanding the system environment:

In order to understand the security requirements of the system, it is important to identify what needs to be protected by analyzing the system environment. The following guidelines are considered in order to study the security needs of the management units:

- The information that is stored and managed by the units.
- Identifying entities within the system that are involved in exchanging information.
- The information that is exchanged between the units and the entities involved.

Each management unit manages and stores information that is required to handle certain services. The USIM unit manages the personal information submitted by the user to obtain a portal account or a user certificate. This information is maintained in an LDAP server that functions as the information repository. The TM unit manages the credential information on behalf of the user. This information demands a high-level of security as a compromise may prove to be detrimental and impact the credibility of the portal service. The RJM unit manages and stores system related information about the resources available on the cluster. This information is stored on an LDAP-based information repository that may be queried by the user by calling the MDS service.

Besides information being stored by each of the management units, there may be situations wherein a function module within a unit needs to obtain information

maintained by a different management unit in order to process the service request. For example, the "Certificate Creation" function module within the TM unit would need access to the user's personal information that is stored in a repository managed by the USIM unit. Although this might involve physical transfer of data over the intranet, it still needs to be secured to prevent any unauthorized access by eavesdroppers on the network. Table 4.2 depicts information exchanged between the different units and entities within the portal system.

Table 4.2 Information exchange between external entities and also within the management units. Each of the management units handle various services independently, but might need access to some information that is maintained by another unit. In such a situation, entities that belong to different units might need to communicate with each other and exchange necessary information.

Interacting Units	Components Involved	Information transferred
Client Browser and the USIM unit	User and webserver	Personal information for portal account and certificate creation, job description, job status and output, etc.
	RRA and webserver	User's account and certificate request information.
USIM and TM units	User Information Repository and Trust Management Server	User's personal information for creation of certificate requests.
	Webserver and the Proxy management server	User's proxy certificate delegated to the webserver upon request.
USIM and RJM units	Webserver and the Gatekeeper	User's proxy certificate and the job description, resource information queried by the user, job status and output, etc.
RJM and TM units	Gatekeeper and the Proxy management server	User's proxy certificate for renewal.

4.4.2 Identifying vulnerabilities and potential threats:

The portal system is exposed to different kinds of security threats and attacks. All communication between the user and the portal system is carried out through a front-end layer presented by the web server. Communication over the Internet is considered unsafe

due to increasing presence of hackers, malicious software, network sniffers, etc. Thus the web server acts as an entry point to the portal not only to legitimate users but also to various malicious users's and programs present on the Internet. Apart from the web server, the portal system also comprises of several key elements like the LDAP directory server, the TMS, the proxy server, the Certificate Authority, etc. which are all connected to the internal network. These service entities also need to be protected against unauthorized access and attacks. The LDAP server within the USIM unit stores user information like the user's password which is used to authenticate legitimate users trying to access the portal. Unauthorized access to this data may lead to an attacker having direct access to the portal and other available resources. On a similar note, unauthorized access to the LDAP Credential Storage server within the TM unit is perceived as a far greater threat to the operation of the portal system as the long-term credentials (certificate and key) of the user are then available to an intruder who may use it to gain access to various resources on the grid and go undetected. Table 4.3 describes some of the most popular and commonly encountered threats together with measures that are taken to counter these threats. A discussion of the design of security mechanisms within the system follows. Implementation details of the security mechanisms used in the design will be presented in the next section.

Table 4.3 Threat Assessment (TA) matrix. The TA matrix describes some of the popular network-based attacks and possible entities within the system that are vulnerable to such attacks. Brief description of each attack is provided to gain an understanding of how each of these threats work and measures that may be taken to safeguard against them.

Threat	Affected entities / service	Description	Solution
DoS attacks	Web server, User Information & Certificate Storage Repositories, Proxy service, CA service, and the Gatekeeper service	DoS attacks target the systems providing some useful service. These attacks severely affect the performance of the service. It is generally caused by submitting malformed requests to the service provider. This attack generally prevents legitimate users from using the service.	DoS attacks can be hard to trace and safeguard against. Monitoring the service and setting up accounting logs could prove to be useful in diagnosing the problem.
Network sniffers	Any communication over the internet or intranet (web server, Storage Repositories, TMS, CA, Proxy server, etc.)	The attacker eavesdrops on the connection between the client and the server, thus gaining unauthorized access to information exchanged by a legitimate client with the server. The attack is passive and user does not change any information that is sniffed.	Strong network security and monitoring policies can help in identifying illegal use of network sniffers. Performing secure and encrypted information exchange between the client and the server also prevents unauthorized access.
Connection hijacking	Occurs when clients communicate over the internet or in the intranet environment (all service providing entities).	The attacker is able to hijack an authorized client session and all connection with the server is then controlled by the attacker. The attacker gains unauthorized access and is able to perform malicious operations with a valid identity.	Setting up connections that are protected by mechanisms like SSL, or TLS help to establish the identity of the client and server entities and thus preventing possibility of such an attack.
Man-in-the middle (replay attacks)	All service entities providing service over the intranet or the internet.	The attacker is positioned between the client and the server and intercepts all unprotected information exchanged by them. The attacker may modify information or steal information like passwords that may be used to access the server and conduct new attacks.	By using encryption mechanisms to protect the information exchanged between the client and the server, unauthorized access may be prevented.

4.4.3 Designing a Security Solution:

Designing a security solution is a two step process which involves describing a Security Policy (SP) [see APPENDIX A] that will define steps to be taken to protect the system from some of the threats that were identified and then designing security mechanisms that will enforce these policies within the system.

The SP specifies the security requirements for the system. The implementation and working of the system must be in accordance with this policy to ensure that the system is protected with the intended level of security. A number of formal models have been proposed to describe the security policies for a system in [BELL73] [CLARK87] [DENN76]. Each of the security models are based on a particular security service that is provided by the system. In order to document a good security policy, the security requirements for the system are specified according to the type of security service being applied. The policy specified is applicable to entities (like the Proxy Server, Certificate and Key management repositories, Certificate Authority) that are built to support the existing security infrastructure (GSI) for grid-enabled resources in the portal environment. Thus, the policy would cover the following areas of security:

- Access control methods applied at each of the entities.
- Information flow control between the managed entities.
- Integrity protection for information managed by the entities.
- Authentication of identities that use the system.

Table 4.4 Access control policy applied to entities within the CSEGrid system. The access policy identifies the subjects (or users) and the objects (or suppliers) of information. The subject and object can be a user, resource or a software process that act upon the information. The relationships between the subjects and objects determine the type of access rights provided and the level of protection applied to access the information.

Item	Description
Subjects	User, RRA, SRA, software agents/process on the Web Server and the TMS
Objects	User Information & Certificate and Key Storage Repositories, Proxy Server, CA, Gatekeeper
Relationships	**User Information Repository:** • Access to the information stored in the repository is restricted to legitimate users, RRA, and the TMS. • Users and the RRA can access information only through a software agent (SA) residing on the web server. • The SA running on behalf of the user is provided read / modify access to his own entry. User may not access another user entry. The access right to modify an entry may be limited to specific attributes within the entry like 'userPassword', 'address', etc. • The SA running on behalf of the RRA is provided read / write / modify / delete access to all user entries with an attribute 'regAuthority' whose value indicates the identity of that particular RRA. They are also provided read / modify access to their respective entries. • The TMS is provided read access to all information pertaining to a user entry, except the 'userPassword' attribute. Only entries that are marked for read by the RRA are available to the TMS. The TMS is responsible for unmarking the entry after reading the information. **Certificate & Key Repository:** • Access to the information stored in the repository is restricted to the SRA and is controlled by a software process residing on the TMS. • The software process is provided read / write / delete access to the entries in the repository. • The SRA software process is also provided read / modify access to the SRA's entry. **Proxy Server:** • Access to user proxy credential repository is restricted to the TMS, the software process on the web server, and the Gatekeeper. • The TMS is provided with read / write access to all credential entries. • The process on the web server and the Gatekeeper are presented read-only access to a particular credential entry identified by the process.

4.4.3.1 Access Control Protection:

Entities within the system provide access to information and resources to services or other entities that handle the user's service request. It is possible to define safe states for the entities within the system by identifying the consumers (or subjects) that need access some information or resource and the suppliers (or objects) that provide the information that may be accessed by the subjects, and then defining a relationship between the subjects and the objects that determines which subject can access an object in what way. Access control is applied using the relationships that are determined. Table 4.4 describes the access control method as applied to some of the entities within the system.

4.4.3.2 Information flow control between entities:

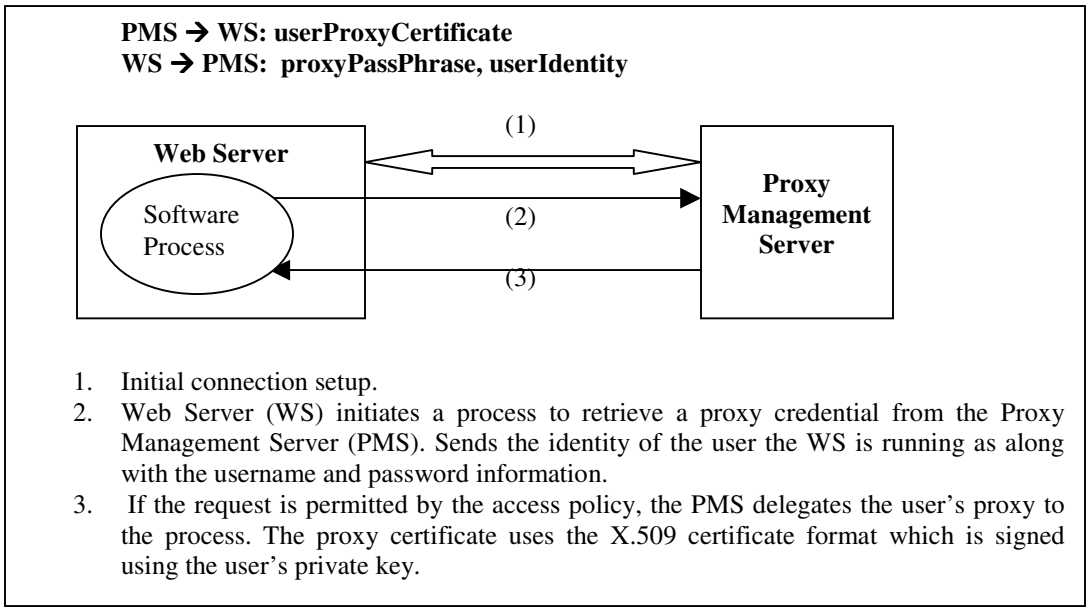

Figure 4.12 Information flow control between entities. This example illustrates the exchange of information between the Web Server (WS) and the Proxy Management Server (PMS) for delegating a user's proxy certificate. The flow relation WS → PMS describes the information that is transferred from the WS to the PMS in the exchange.

Another important security measure is to outline the information that is transferred between the entities (or objects) within the system. The concept of flow relations [DENN76] can be applied at an abstract level to represent all valid information exchange that takes place within the system. A flow relation A → B is defined between to entities 'A' and 'B' and occurs when a set of operations performed at 'A' results in information being transferred from 'A' to entity 'B'. Here the symbol "→" is used to represent the flow relation. Together with the access control applied by the entities, the flow relation describes all valid information exchange that occurs within the system. The flow relations serve as a guideline for implementing the software operations that result in information transfer that are deemed valid by the flow relations. Figure 4.12 identifies a simple flow relation between the Web Server and the Proxy Management Server.

4.4.3.3 Integrity Protection:

Protecting the information managed by storage entities within the system from being corrupted or modified to reflect illegal values is essential from a security standpoint. Here, integrity of information would mean preserving the information from unwanted changes thereby affecting the normal operation of the system. The access control and information control policy prove to be helpful in restricting the subjects that have access to the information and the type of information they have access to. A set of integrity requirements may be specified that are applied to all operations that perform changes to information managed by an entity. The integrity check provides protection from accidental as well as intentional attempts of modifying information.

4.4.3.4 Authentication Schemes:

Authentication is the means by which the identity of a subject and an object may be verified. The choice of an authentication mode depends on:

- The sensitivity of the information or resource that is to be protected.
- The service type and the environment in which the service is provided (over a secure network or an unprotected network).
- The target base of the users.

The CSEGrid portal system employs various authentication schemes to validate the identity of the user (or processes acting on behalf of a user). For instance, the portal uses a password-based authentication mechanism to identify the users accessing the portal services. As mentioned earlier, the grid resources that are built using Globus middlware employ the X.509 based certificate authentication mechanism to verify the identity of entities on the grid. The Grid Security Infrastructure (GSI) component of the Globus software is responsible for implementing the authentication mechanism. It also supports additional features like single sign-on (SSO) and certificate delegation that are particularly useful in the grid environment. The CSEGrid portal system is designed to provide a certificate credential management service for the grid user. In this scheme, the user's credentials (certificate and private key associated with the certificate) are stored in a secure credential repository. Thus, suitable authentication mechanisms need to be designed to safeguard these credentials and allow an authorized user (SRA) to be able to retrieve them whenever required.

In this section, we described the architecture of the CSEGrid portal system followed by a discussion of the design components within the various management units. To conclude the section, security considerations in designing the management units were outlined. The following section describes the implementation details of the system and the security mechanisms employed in securing the system.

CHAPTER 5

IMPLEMENTATION

The architecture of the CSEGrid portal system (as seen earlier in figure 4.1) is categorized in the form of various management units. The design follows the popular n-tier web application architecture by partitioning the system into various units based on the functionality handled by the units. This makes implementation of the various services relatively easier and straightforward.

The development of the portal system is carried out in stages. Each stage comprises of the following steps: construction of the management unit infrastructure, installation and configuration of the software components specified in the design and finally implementation of the essential services provided by the unit. One of the design goals is to provide adequate security within the system. The implementation of the system realizes this goal by securing the services and infrastructure at each stage. The portal system makes use of existing web and grid technologies that meet the requirements of the system. A brief description of these technologies is presented for a better understanding.

5.1 Supported Technologies:

Globus

Globus is a middleware technology that is used for building the grid infrastructure. It offers implementation of services like data management, resource discovery and monitoring, job submission and management, and security services. The Globus project has defined grid protocols like GridFTP, MDS, and GRAM that implement the grid services. The services that are implemented using these protocols are provided in the form of software packages and libraries which constitute the Globus Toolkit.

MyProxy

MyProxy is an online credential repository system that allows users to store short-term certificate credentials on the proxy server. These credentials may be retrieved at a later time by the user or an application on the user's behalf. MyProxy was primarily designed to address the problem faced by applications that require authenticating the user to remote resources in order to gain access to the services provided by them. MyProxy allows these applications to retrieve a delegated proxy credential for the user that has been previously stored on the proxy server. The proxy service is ideally suited in the web environment, as the user's web browser does not support delegation of credentials.

GridPort

GridPort [GRIDPORT] is a Commodity Grid Toolkit [see APPENDIX A] software implemented as a set of libraries written in the Perl [PERL] language. GridPort supports rapid development of web-based grid applications by providing access to basic grid services like grid authentication, job submission and management, resource monitoring, and data transfer that are available on grid resources. The toolkit also provides additional features like session and account management for grid portals.

LDAP Directory Service

The LDAP directory service provides a simple, lightweight, cost-effective, and easy to implement solution for data management requirements. The directory server operates on a client-server model and supports LDAP protocol operations to support communication between the client and the server. Information within the directory server is organized in the form of an inverted tree (also referred to as Directory Information Tree). The LDAP service provides features like flexibility in the type of information stored, scalability, data replication, security, and others that make it an attractive choice for data storage. The CSEGrid portal system uses OpenLDAP [OPENLDAP], a popular open source directory service implementation.

Apache

The Apache HTTP server is a popular open source implementation of web services used on the Internet. The portal system makes use of a secure version of the

HTTP protocol, called HTTPS, for communication between client browsers and the web server. The Apache web server software is flexible and allows additional features to be incorporated in the form of dynamic modules. For example, a module named mod_ssl [MODSSL] may be used to build SSL functionality into the web server software.

5.2 Implementation of the Portal Units:

The implementation of the CSEGrid portal system prototype was carried out on a test-bed comprising of four Intel Pentium-III PCs with 933 MHz processor and 128MB memory. Each of these machines was installed with Fedora Core 1.0 Linux operating system. Implementation of individual management units is as described below:

5.2.1 User and Service Interface Management Unit:

The portal system uses the Apache web server software as the front end to serve user requests. Users may use any standard web browser (MS Internet Explorer 4.0 or greater, Netscape, etc.) to access the portal interfaces. Communication between the client browser and the web server is secured by enabling SSL support on the server side and is carried out using the HTTPS protocol. Access to the services provided by the portal system is restricted to authenticated users. All interaction between the user and the portal is carried out through interfaces that are implemented as HTML forms. The user fills out the form with required parameters which is then submitted by the client browser to the web server. The input provided by the user is then passed onto the function handler which processes the information and carries out the requested operation. The function handling

modules are implemented as CGI/Perl scripts which are invoked by the web server based on the service requested for by the user.

5.2.1.1 Handling Grid Service Requests

The portal system provides grid services to the user by presenting UIs that may be used to submit relevant information to execute the service. The appropriate function handler script uses the GridPort middleware software to interface with the lower level grid resource and submit the request. For example, if the user wants to execute a batch job, he submits the location of the file that contains the job script in the request. The function handler script that handles job requests is invoked when the user submits the request. This script uses the *Cog::Globus::Job* module, creates a new job object with an associated job-id. It then invokes the *submit* method on the job object to send the job request to the remote gatekeeper listening for user requests. Figure 5.1 shows the processing of the job request as discussed in this example.

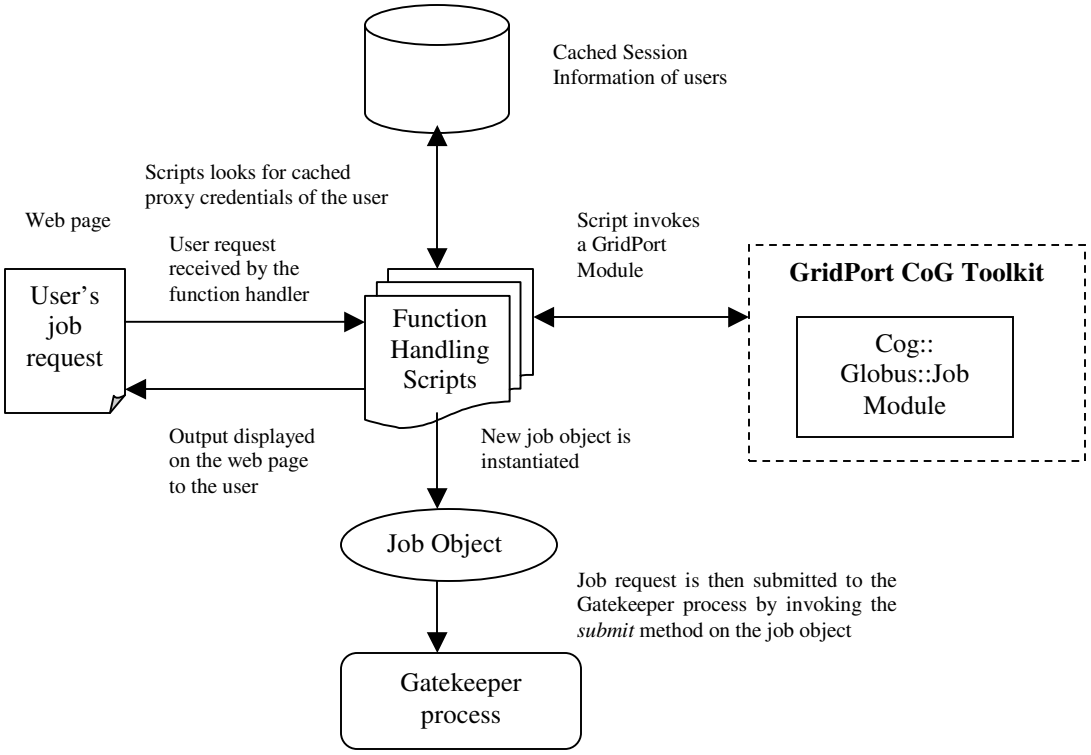

Figure 5.1 Processing of a service request within the USIM unit. The example shown in the figure indicates the handling of a user request for execution of a batch job on one of the remote grid nodes. The function handler utilizes the GridPort toolkit services to interface with low-level grid services.

In the above example, it is assumed that the user who submitted the request has a valid proxy credential stored on the PMS. The credential is required to identify the user to the remote grid entity. Each function handler that processes a grid service request verifies if the user has a valid proxy credential available prior to submitting the request. If this credential is not available, then the function handler invokes the Session Authentication function handler to obtain the Myproxy passphrase from the user and retrieve a delegated user proxy from the MyProxy repository.

5.2.1.2 Session Management

Portal services are protected from unauthorized access by authenticating users prior to processing their request. But with HTTP being a stateless protocol, the web server is unable to keep track of the user's identity between consecutive requests. This would mean the user would have to authenticate to the web server with each request which is not desirable. In order to provide convenient access to the portal a suitable state management mechanism must be implemented that will keep track of all activities of the user from the time he or she first logs into the portal until the user decides to leave the portal. This period is usually referred to as a user session. Each session is associated with some relevant information like the cached credentials of the user which is used during the session, *jobID* of a previously submitted job, among others. Each session is identified by a unique *sessionID* which is a large random number generated by the function handler at the start of the user's session. The *sessionID* is saved in a temporary file on the web server and is transmitted between consecutive requests from the client browser. No other session information is exchanged between the browser and the web server for security reasons. The choice of a large random number makes forging the *sessionID* difficult if not impossible for users with malicious intent. Sessions are terminated when the user logs out of the portal system. In the event of an abnormal termination like loss of network connectivity between the user's web browser and the portal server, it becomes necessary to expire the user's session in order to protect against intruders looking to gain access to the system. This is achieved by associating a timeout period with the user's session.

5.2.1.3 Portal Accounts and User Authentication

In order to access any of the services of the portal system, the user is required to obtain an account on the system. The process of obtaining an account was shown through a work-flow diagram in the design section (as seen earlier in figure 4.5). Users approved to obtain an account are provided with a userid and a password that may be used to authenticate to the portal. This information along with some other personal information about the user is stored in the LDAP information repository. The LDAP repository server is implemented using OpenLDAP software. The backend storage used by OpenLDAP in our configuration is Sleepycat Software's BerkeleyDB [BERKELEYDB]. The LDAP server is configured along with additional software libraries like Transport Layer Security (TLS) [OPENSSL] in order to support encrypted communication between the client and the LDAP server.

The LDAP repository server is configured to support mutual authentication between the LDAP client (which in this case would be the web server) and the LDAP repository server. This allows the client and the LDAP server to ensure they are communicating with the entity specified in the certificate that they obtain from each other. The authentication process is carried out using the SSL Handshake protocol [RFC2246]. It is assumed that both the client and the server possess a certificate that has been signed by the CA. The client and the server need to also have the certificate of the CA that signed the certificate of the opposite entity (which in our case happens to be the same CA) and they must be able to trust the CA that signed their certificates.

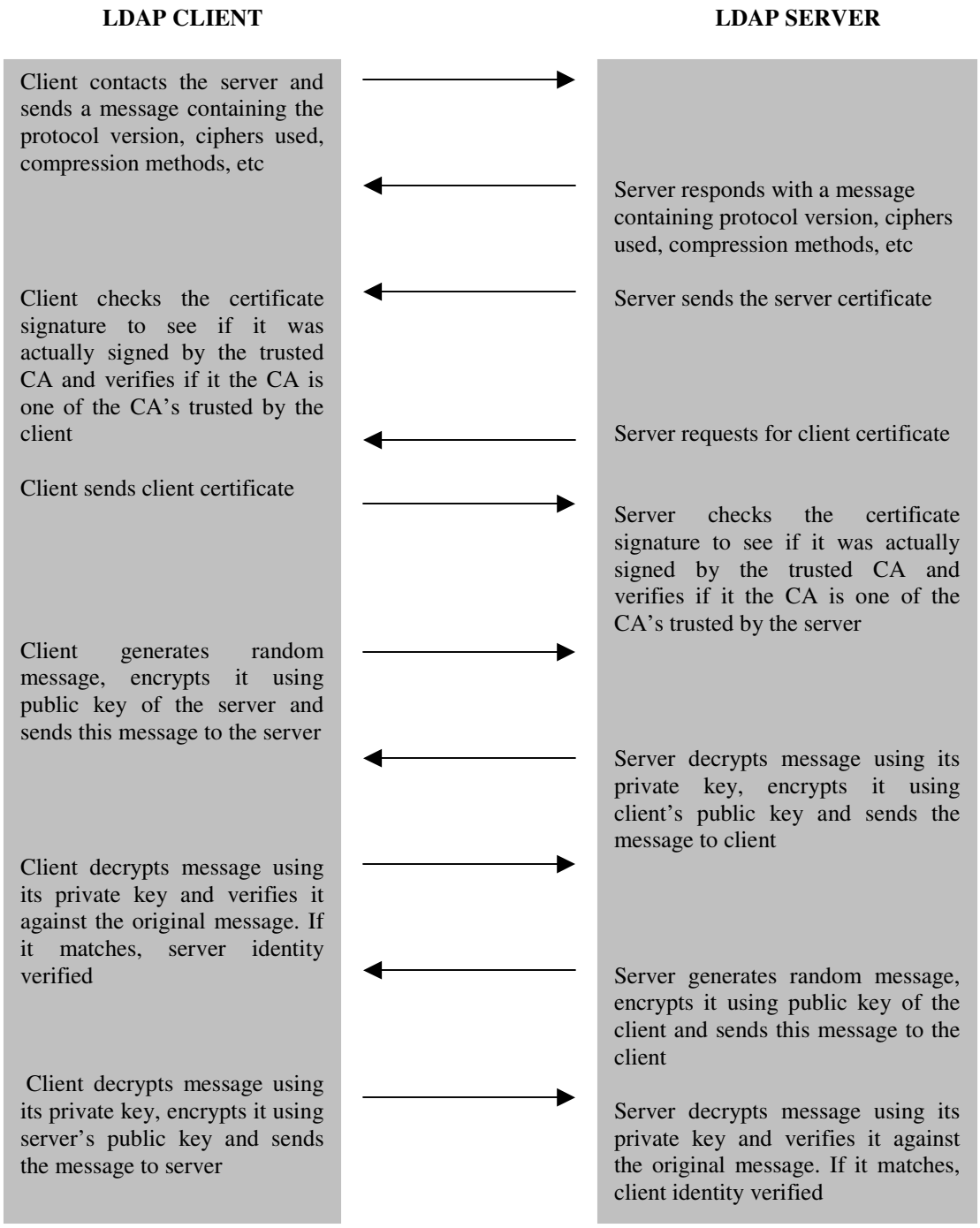

Figure 5.2 The SSL handshake protocol [RFC2246]. The figure shows the sequence of messages exchanged between the client and the server. The direction of the arrow heads indicates the flow of messages between the authenticating entities.

Figure 5.2 describes the authentication process between an LDAP client service and the server using SSL Handshake.

The OpenLDAP software allows us to specify a strong access control policy for the information stored on the LDAP server through the server configuration file.

5.2.2 Trust Management Unit:

The implementation of the TM unit involves setting up of a Trust Management service that provides a centralized storage mechanism for the user's certificate credentials. Setting up of a centralized credential management service raises many security and operational concerns such as:

- Compromise of the centralized credential repository would in effect compromise all the certificates stored in the repository.
- A centralized repository becomes an obvious target for attackers that wish to disrupt the service.
- By providing a common storage area for both the certificate and private key, there is a greater risk of a user identity being stolen as an intruder now has access to all information that is needed to forge the identity of the user.
- A single repository for storage would mean the service must be available at all times and no downtime would be tolerated.

The TM unit seeks to counter most of the prevailing vulnerabilities of centralized credential storage by implementing a strong security policy. The credential repository is implemented using a LDAP server which allows us to specify a fine-grained access

control policy, conveniently store the user certificate and private keys in a binary form supported by the software, and implement a simple and yet powerful replication scheme to backup the repository. The certificate and private key information are placed on the same server, but are controlled by implementing different access control levels. The private key is made accessible only to the SRA who requires the key to generate proxy credentials on the user's behalf. Access to the certificate is mediated through a function handler and is accessible by the SRA. The LDAP repository server makes use of firewalls and / or TCP wrappers [TCPW] to restrict access to the server based on the IP address of the client. Although this security measure does not prevent an attacker from spoofing the IP address of the source machine, it still places adequate restrictions to discourage such an attack. Replication of the repository offers a number of benefits like:

- The backup server helps to provide a reliable credential storage service. The backup server may replace the primary server in the event of a crash or during scheduled downtimes of the primary server.
- The backup server may be used to compare with information stored on the primary server in situations where user's private key is suspected to have been modified or some other unauthorized changes have been made. This helps to restore the sanity of the service.

A simple replication scheme of the repository server is shown in figure 5.3.

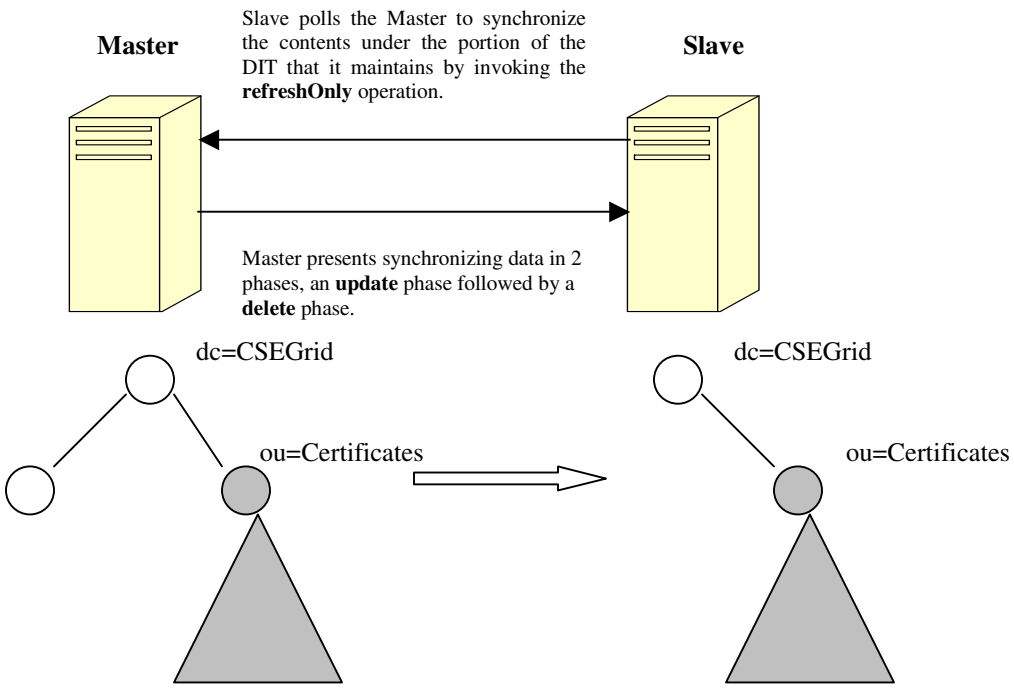

Figure 5.3 LDAP directory replication scheme. The replication scheme shown involves one master and one slave node. The master node is made accessible to the client services with access control restrictions applied to the entries within the DIT. The slave node is used as a backup for the information stored on the master node and hence is not made available to the clients. As shown in the figure, only part of the subtree (indicated by the shaded region) is replicated onto the slave. The subtree corresponds to all entries containing the user's long-term certificate credentials. Access control policies applied to entries within this subtree are preserved on the replicated copy.

The replication scheme is implemented using a slave-side replication engine called *syncrepl* [OPENLDAP]. Only part of the DIT on the Master server that contains entries corresponding to the user's credential information is replicated onto the slave. Synchronization of this information is performed using a pull-method wherein the slave polls the master server requesting for all updated information at some regular intervals. This is achieved with the slave invoking the *refreshOnly* synchronization operation in the polling request. The master performs this operation in two stages, the first being an *update phase* wherein it sends all attributes of the updated entries within the scope of

synchronization specified in the configuration. This is followed by the *delete phase* in which the master sends delete messages that contain the name of the entry and a synchronization control for each entry that is deleted. The slave makes changes to the replica by looking at the synchronization control which indicates the state of the entry.

Apart from the certificate storage component, the design of the TM unit contains additional components that provide different services to the portal user. The Trust Management Server is an authoritative server that controls access to other components within the TM unit. Operations like creation of user CSR's and proxy credentials, retrieving signed certificates from the CA are all initiated on the TMS by the Site Management Authority (SRA). The SRA is a person authorized to carry out these operations after verifying the validity of requests received from the user. The SRA is provided access to these operations through a secure command line tool like the SSH program. The function handlers that execute the operations are implemented in the form of scripts that are written in the Perl language.

For security reasons, the SRA is required to authenticate with the services in order to perform the operations. For example, in order to store a proxy credential on the Proxy Management Sever, the SRA needs to first authenticate with the TMS to gain access to the scripts used to create and delegate the proxy credentials to the PMS. Then, the SRA is required to present its identity along with the user pass phrase information in order to retrieve a copy of the user's certificate from the repository. The proxy credential is then generated and then sent to the PMS along with the SRA's credentials. The SRA

must be authorized to perform this action by the PMS. This is done by specifying the DN name of the SRA in the MyProxy server configuration file.

The portal system implements a self-signed certificate authority (CA) that issues certificates to hosts and users that are part of the system. The responsibility of verifying the identity of a user requesting a certificate is delegated to the RRA designated for the site or institution with which the user is associated and the SRA of the portal site. The CA service is based on the OpenSSL implementation that uses the X.509 certificate format. The CA verifies the information within the CSR and then signs it using the CA private key. The signed certificates are retrieved by the SRA to be stored in the credential repository. A secure copy (*scp*) or secure FTP protocol may be used to perform the transfer.

5.2.3 Resource and Job Management Unit:

The implementation of the RJM unit involves setting up of a grid environment that is made accessible to the user through the portal. The Globus Toolkit (version 2.4) software is used to build the grid environment on the testbed. The setup includes one node running the Gatekeeper service and two computing nodes on which the user's jobs are executed.

The Gatekeeper node acts as a gateway for all grid service and resource requests. A gatekeeper process is started on this gateway node which listens for requests from the portal server. The Gatekeeper relies on GSI authentication to verify the identity of the client requesting the service. GSI authentication requires both parties to authenticate with

each other. The client process which is acting on behalf of the user, then submits the proxy credential of the user to the Gatekeeper. The Gatekeeper checks to see if the CA identified on the proxy credential is one that it trusts. It then verifies to see if the DN name of the user specified on the proxy certificate is acceptable by looking at the signing policy file (*/etc/grid-security/certificate/<CA_Name_Hash>.signing_policy).* If the certificate is found to be valid, the Gatekeeper checks to see if the subject in the certificate is authorized to access the grid services / resources by looking in the */etc/grid-security/grid-map-file* which contains mapping between a user's global account and a local account on the grid resources. It is assumed that the local accounts are setup on the grid resources based on a policy specified by the site administrator.

The compute nodes are configured to support the following Globus grid services:

- Resource Management Service using GRAM.
- Resource Discovery and Information Service using MDS.
- Data Management Service using GridFTP.

Apart from these services that are configured, the GSI security service is also set up to manage the security requirements of the grid resources.

CHAPTER 6

CASE STUDY AND EVALUATION

In the earlier sections we have seen the steps involved in the construction of the prototype of the CSEGrid portal system. The proposed scheme addresses the need for a credential management service to provide a secure ubiquitous grid computing environment to the user. This section is divided into a discussion of how the CSEGrid portal system fits into an example grid environment to provide users with transparent access to services in a secure fashion and then an evaluation of the portal system with respect to other grid portal system that are currently being used by other organizations.

6.1 CSEGrid Portal System – The Big Picture

Figure 6.1 shows a sample setting for the CSEGrid portal system comprising of three organizations. The grid environment consists of various heterogeneous resources available at the resource sites at organizations '1' and '2'. The users are associated with various research groups or departments within an organization. The resource providers' viz. resource sites 'A', 'B', 'C' and the users combine to form what is referred to as a virtual organization (VO). The CSEGrid portal system offers a simple and feasible security solution to setup access to grid resources and/or services for members within the VO. These members delegate trust to the portal system to manage the credentials that are

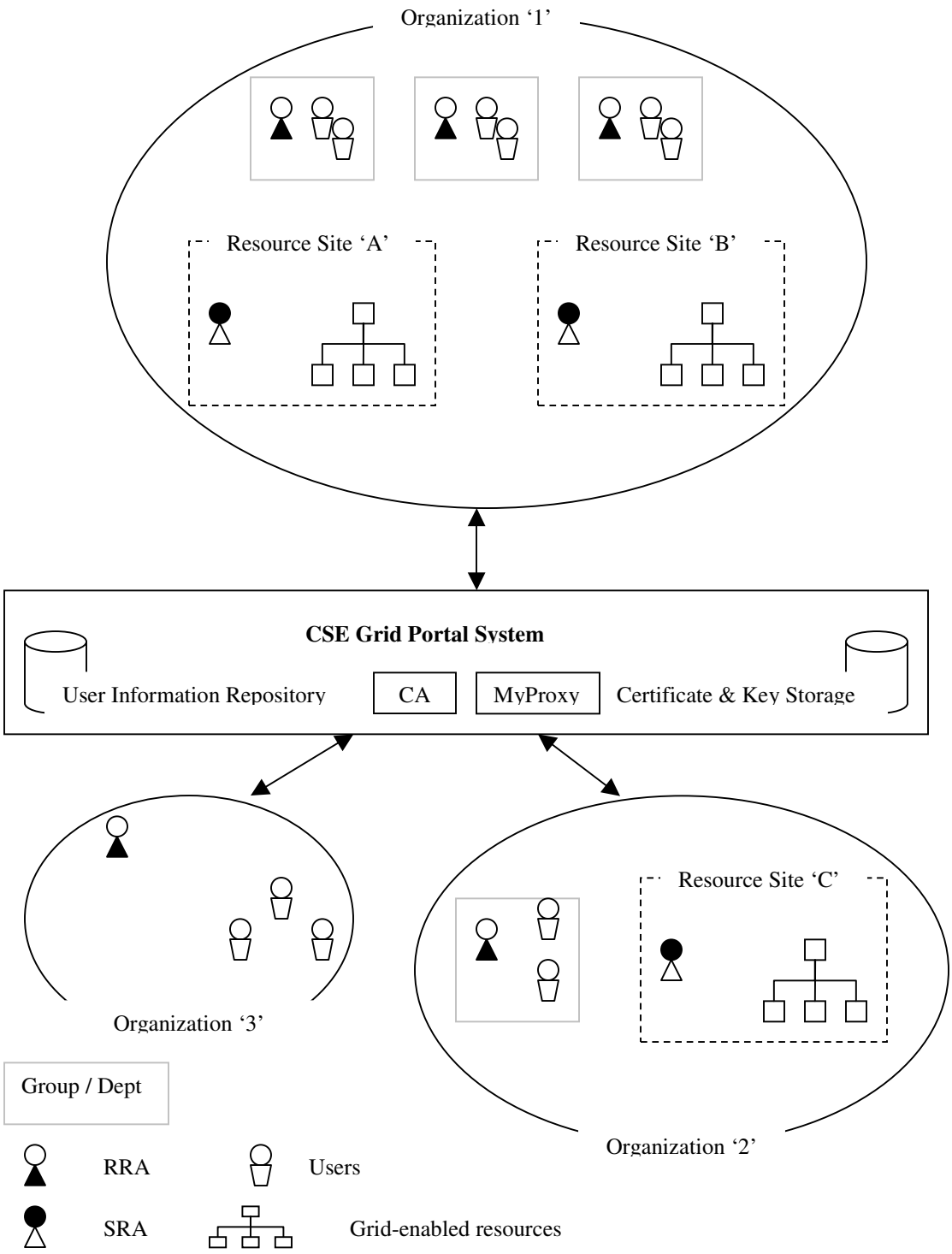

Figure 6.1 CSEGrid portal system in a sample grid environment.

required by the portal server to authenticate the user in the grid environment. The CSEGrid portal system relies on setting up of a chain of trust, as shown in figure 6.2, to provide the credential management service for the users.

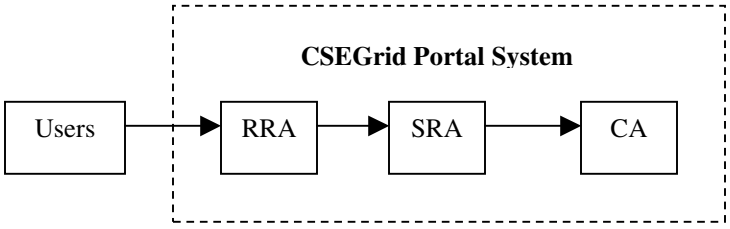

Figure 6.2 Trust relationship employed by the CSEGrid portal system. The direction of the arrows indicates flow of trust information that is sought by the entity indicated by the arrow-head to provide the trust management service to the user.

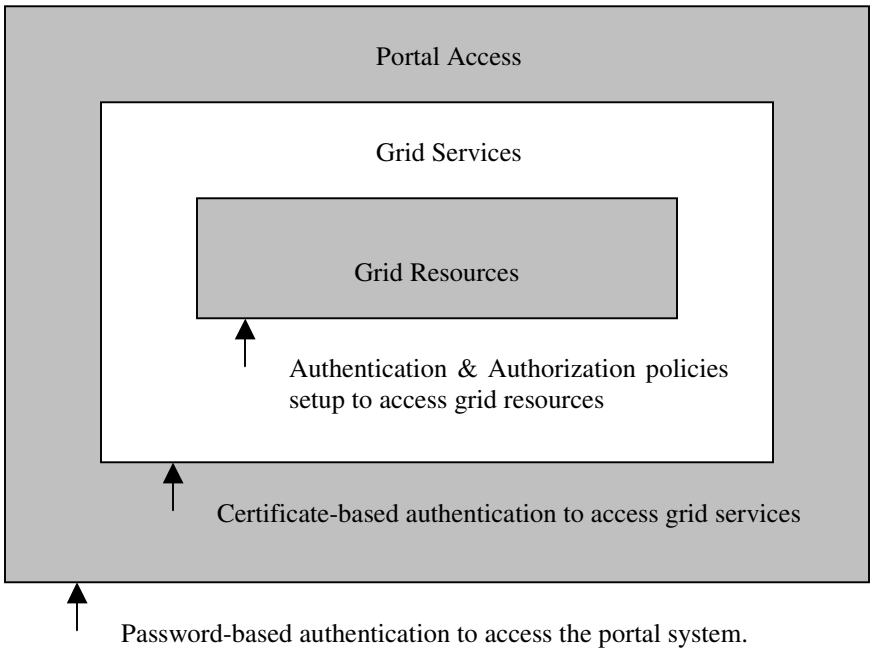

Figure 6.3 CSEGrid implements a layered security solution. This restricts the access available to the user at each layer and provides stronger security to the services and resources against unauthorized access.

The CSEGrid takes a layered approach for providing secure access to the services and resources. Figure 6.3 shows how each layer is protected by a certain security mechanism.

6.1.1 Identifying the Trust Entities:

The first step in setting up the trust management service is to identify the trust entities within the VO. The portal system requires setting up a chain of trust between the trust entities. This trusted-chain is represented in a hierarchical form with the Certificate Authority (CA) forming the root trust entity within the system. The CA signs the certificates which represent the identity of the user in the grid environment. The CA only signs requests received from authorized SRA identified within the VO. The SRA represent the stakeholders within the system. In this case, each resource site is represented by a SRA who is responsible for credential management operations on behalf of users who are allowed to access resources on that site. Thus, the site SRA does not provide trust management service to any user that is not allowed access to that particular site. Some of the credential management operations performed by the SRA on behalf of the user are as follows:

- Creation of certificate signing requests.
- Storing the user's signed certificate and key information in a secure repository.
- Delegating a proxy certificate to the Proxy server.

The SRA relies on the RRA to verify the identity of the users and to ensure that they belong to groups or departments that the site is willing to provide access to. The RRA is a person authorized within a group or department to verify the identity of the user

and approve their requests if they meet the required level of assurance specified by the VO policy.

6.2.2 Providing the trust service:

As an example, we consider a situation wherein a user within the VO is trying to gain access to services provided by the CSEGrid portal system for the first time (see figure 6.4).

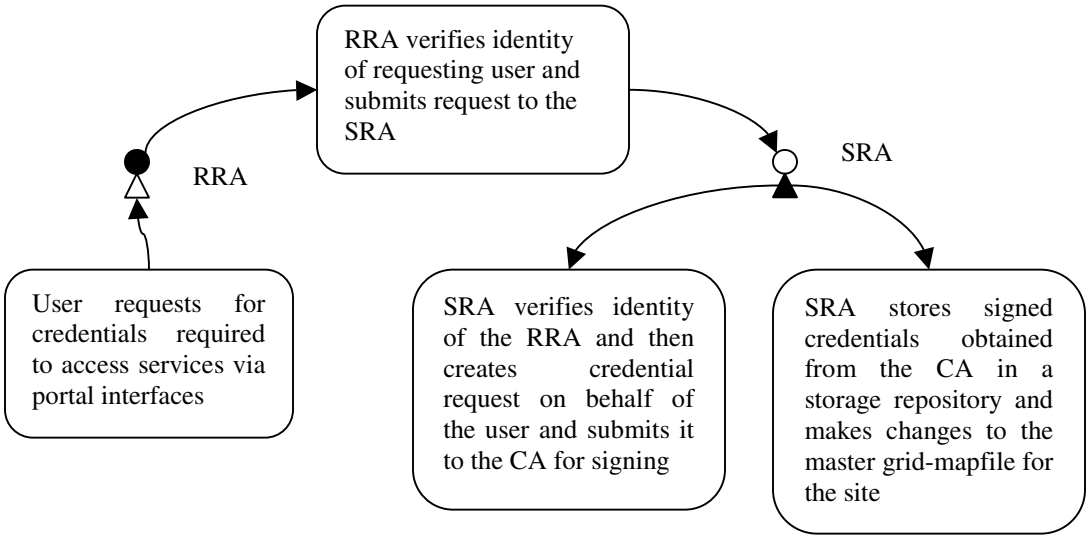

Figure 6.4 An example of setting up user certificate on the portal system. The figure shows the steps involved in certificate creation process and the trust entities that initiate those steps within the portal system.

The CSEGrid portal system thus provides a trust model wherein the trusted entities are first identified and then trust relations are established between these entities according to the policy specified by the VO. This model provides for better accounting and non-repudiation [see APPENDIX A] of the actions taken within the portal system for providing a credential management service to the user of the portal system.

6.2 Evaluating the Portal System

The CSEGrid portal system has been designed to provide users with a secure anytime and anywhere access to resources and services available to them as part of the VO. Security requirements tend to place restrictions in making the grid services available in a web portal environment. The trust management service helps to resolve some of the restrictions that are generally faced using other portal implementations by allowing the user's credential to be stored and managed by the portal system. We evaluate the CSEGrid portal system against some of the other portal implementations that we studied. The observations made in this comparative study are described in the following discussion:

1) Lightweight portal implementation: The portal design ensures that the implementation of the front end server is relatively lightweight. The user interfaces are presented in simple HTML format and the function handlers are implemented using Perl/CGI. This lightweight solution allows rapid development and deployment of the portal system which suits the portal system requirements. Moreover this solution ensures that the UIs are available to the users through almost any client web browser which may or may not support features like XML support, running Java applets, etc that are required by other portals in rendering the user interfaces.
2) Trust-based model The CSEGrid portal like others implementations of grid portal systems that we examined incorporates the existing grid security technology (GSI) into its design. The GSI technology is still in the evolution stage, but has

become very popular and is accepted as a working standard among the grid community. GSI uses the X.509 certificates for verifying the grid identity of users and resources.

In order to support GSI in a grid portal environment, existing portal implementations consider trust relationships between the user and the certificate issuing authority (CA), whereas the CSEGrid portal system employs the trusted-chain model. The trust model realizes the need for better security management of the user's credentials. The issuance and revocation of certificate is based on the trust existing between the SRA and the CA rather than between the user and the CA. The traditional trust model employed by other portal implementations fails to provide immunity against compromise of users credential due to mismanagement or in the event of the user no longer being part of the organization that he was associated with. If the user fails to report the changes to the CA, his credentials may be used by an intruder to gain unauthorized access to the portal system. In the CSEGrid system the SRA relies on the RRA to provide accurate information regarding the status of the user account. If the user leaves the organization and does not need access to the portal system, the RRA terminates the portal account and informs the SRA about the status of the user. As the user's credential will no longer be required, the SRA destroys the user's credential information that it maintains and requests the CA to revoke the user's certificate.

3) Information Management: The CSEGrid portal system employs the LDAP directory service for its data management needs over relational databases that are

generally being employed by other portal implementations. The features that influenced our decision were discussed in the design section of this paper. The performance costs are far lower as the LDAP server handles requests 10 to 100 times greater than relational databases. This ensures that the performance of the portal remains fairly consistent in the event of large number of users accessing the portal simultaneously.

4) Credential Management: Most portal implementations expect the user to obtain certificates from the CA setup by the portal system and hold the user responsible to safeguard those credentials. The user is then required to delegate a short-term proxy certificate to a proxy server that is used by the portal system through some out-of-band means. This may be cumbersome and may also result in the long-term credential of the user being compromised if the user fails to take adequate measures to safeguard the credentials. The user has to ensure that he has a proxy delegated to the proxy server at all times so that he may be able to access the portal service uninterrupted. This might be a problem, for instance, if the user were to be accessing the portal from a remote location where his long-term credentials were not available to him and during this period his proxy credential was to expire. The user would then be left with no access to the portal until he once again sets up a new valid proxy on the server.

The CSEGrid portal system addresses these problems by providing a novel credential management service. The credential management service maintains the user's credentials

for a period equal to the lifetime of the credential. The credentials are stored in a repository with access restricted to only authorized users. In our design, only the portal server needs to be authorized to retrieve proxy credentials from the proxy server. In the event of expiry of the proxy credential, the user simply requests for a new proxy certificate to be delegated to the proxy server using the portal interface. An alternative would be to send an email to notify the user about an expiring proxy so that the user may take necessary action.

As part of the evaluation of the CSEGrid portal system, we examine the implementation choices made for building this portal and compare them with other known grid portal implementations. The following study describes the considerations made while selecting the various technologies in building the CSEGrid system.

6.2.1 Portal Server Implementation:

The CSEGrid portal system like other grid portals is essentially based on the 3-tier architecture. The portal service is built using a web server that establishes communication with client computer over the web and various application services that define the functionality of the portal. The application services may reside locally on the web server or on separate machines that can communicate with the web server.

The Apache [APACHE] web server is the most popular web server software used in most of the present grid portal systems. Apache web server has proved to be one of the most powerful UNIX-based web servers providing high performance, more functionality

through the configurable options, and availability of source code for custom configuration.

Nanda et al. [NANDA99] in their paper on performance study of Apache web server have identified potential bottlenecks that affect the performance of the server. Some of the main factors that were identified as potential causes for the performance slowdown are:

- *Context switching:* The Apache server is based on a multi-process architecture in which multiple client requests are handled concurrently by individual processes. The server forks a new process every time a new request is received and which cannot be handled by the existing processes in the process pool. The context switches between these processes incurs a certain amount of delay on the server which could affect the performance of the server. It is thus important to keep in mind the effects of context switching while configuring the maximum number of processes that can be forked by the server. It has been found that web servers based on multi-threaded architectures incur lower overheads on context switches [NANDA99].
- *RAM capacity:* The size of memory affects the amount of disk caching that is performed by the server. If the size of the RAM is small, the CPU utilization of the server falls as it waits to perform disk I/O in order to fetch the files into memory. Thus, it is important to have sufficient memory of the server that provides the web service and must be decided based on the anticipated load.

- *Size of client request:* The size of the web requests affects the behavior of the web server and the system performance. For smaller requests, the server CPU time in reading or writing data from the disk and transferring the data onto the network. While for larger requests, the server is busy handling network activity. This observation in [NANDA99] leads us to the need for caching data on the web server which reduces the disk I/O and hence improving performance. This is especially important in the context on a portal server that needs to generate dynamic output based on the user requests.

Most of the grid portals use the Apache web server for handling web requests from users seeking access to portal services. Some of the common implementation features among the grid portals are:

- Secure web transactions using SSL via HTTPS to protect information served to the users.
- Implementation of additional security features to protect access to portal services.
- Integration of the web server with the application layer that handles the requests for various portal services.

The CSEGrid portal chooses the Apache web server to host the portal services. Some of the factors that influenced our choice were the feature rich configuration options of the web server, stable performance, and the flexibility in terms of being able to include new modules and extension web APIs.

6.2.2 Security Implementation:

In providing the portal service it is necessary to emphasize the need for protecting information exchanged between the user and the resources with which they wish to communicate with. Understanding this need, most of the grid portals that were designed have employed different methods to provide secure service.

Some of the security considerations that are common to these portal systems are listed below:

- Providing secure web communication using the HTTPS protocol.
- Restricting portal access to authenticated users thereby securing portal services from unsuspecting intruders. To accomplish this portals employ a password-based authentication scheme.
- Employing existing X.509 certificate-based authentication in securing access to back-end grid resources.

It has been observed that the grid portals provide a two-level security for communication between the client browser and the portal server and communication between the portal services with low-level grid services. By providing this layered security approach, the portal systems which are based on a 3-tier web architecture are able to ensure that communication between each of the elements within different layers is protected from intruders both within and outside their network.

The CSEGrid portal system presents a hierarchical trust model to extend the existing security solution between the middle layer which comprises of the portal server and the back-end layer which is made of the various grid resources. A sample grid

scenario was presented earlier in this section where this model was applied. Some of the benefits of applying this model within a grid scenario that comprises of various Virtual Organizations (VO's) are:

- Integrating a new site (VO) into the grid portal environment is simple. This is possible by identifying suitable trust entities within the VO's to which the Certification Authority (CA) can delegate the trust to.
- Provides for better accountability through established trust relationships between the trusted entities while issuing grid certificates to end users.
- Ability for users from different sites to maintain multiple certificate credentials with the portal server as long as the site is identified by trusted entities that are part of the same hierarchy.

6.2.3 Portal Service Implementation:

The portal services are accessed using any of the standard browsers available on the user's computer. Information exchange between the client browser and the portal server is carried out using HTTPS, a secure version of the HTTP standard used for web communication. The services provided by the portal enable users to request information that is obtained from the lower-level grid resources. Thus, the portal server is required to dynamically generate web pages in a suitable format which allows the user to view the information in the client browser. A number of web technologies are available today that allow creation of web pages in real-time. A comparative study of these technologies was made while making a choice for the CSEGrid implementation. We studied some of the

popular ones that are currently being used in other grid-portal projects and made our choice based on this study and our portal requirements. In order to perform a comparison we need to identify suitable metrics that allows us to qualify a particular technology to meet the portal implementation requirements.

- *Throughput:* The throughput of the system is measured as a ratio of the number of simultaneous client requests that are processed by the server in a given time interval to the number of clients making these requests.
- *Support for concurrent client requests:* It is important that the implemented solution is able to offer multiple clients to make requests for resources at the same time. The solution should be able to support large number of concurrent users without any appreciable increase in the Response rate and decrease in throughput.
- *Response rate:* The response rate is the number of client requests that can be processed by the server in a given period of time. This function depends on the response time for a given request which is the amount of time it takes for the client to receive a response back from the server from the time the client first makes a request, and also on the CPU utilization of the server.
- *Request capacity:* This is the maximum number of requests that can be handled by the server at any specified time.
- *CPU utilization on the server:* The CPU utilization is measured to be the amount of CPU cycles that are utilized in processing the requests.

In order to choose a suitable web technology to implement the portal services, we looked at some of the current web methodologies that are extensively used by various organizations and research groups.

Common Gateway Interface (CGI) is one of the traditional ways of authoring web applications. It is a simple programming interface which is supported by most web servers. CGI programs may be developed in a variety of languages like C, Perl, C++, etc.

FastCGI [FAST-CGI] is a superior, open web server interface that was implemented by Open Market to address the inherent issues with the commonly used CGI technology. We will see a more detailed comparison in the discussion to follow.

Java Servlets are server-side Java code that is executed on the server to process client requests. These servlets execute within a servlet container that contain pre-loaded classes implemented in Java programming language.

Table 6.1 presents an evaluation of features provided by each of the above mentioned development solutions.

Table 6.1: Comparison chart of some of the salient features of web development.

Feature	CGI	FastCGI	Servlets
Request Handling	CGI creates a new process for each client request that is received.	FastCGI provide persistent processes that may be reused to handle multiple client requests.	Each new client request spawns a new thread that handles the request independent of the other threads.

Table 6.1 – *Continued*.

Session Tracking	This feature is not built-into CGI. Additional libraries that provide this feature can be incorporated to generate objects that maintain session information.	FastCGI incorporates a feature that routes requests received from the same client to the same server. This feature is called session affinity.	The Servlet API provides support for saving state information in the form of cookies and session objects. The API allows creation of session objects for the period the user interacts with the server. These objects are destroyed once the connection between the client browser and the server is closed.
Performance Overhead	The inefficient in handling concurrent client requests. CGI programs spawn a separate child process for each new client request. Spawning a new process for every request proves to be expensive as the OS has to load the program, allocate memory and again unload the program and deallocate the memory for every process. This result in a large overhead due to the amount of context switching that the OS needs to perform so that multiple processes may be handled concurrently.	FastCGI performs much better than traditional CGI as it reuses child processes from a connection pool. Every time the portal server receives a new request, one of the available FastCGI processes establishes connections with the server and obtains environment variables and other information from the server. Once the request is processed and output sent back to the server, the connection is closed and the process returns back into the pool without exiting.	Servlets provide substantial performance benefit owing to its multi-threaded architecture. Multiple servlets can be run in parallel within the same process on the server by using separate threads which handle each of the individual requests. The use of light-weight threads incurs low context-switching costs.
Process Isolation	CGI applications run as separate processes and thus prevent rogue applications from crashing the entire server.	FastCGI also benefits from process isolation in keeping malicious applications from stalling the web service. The individual child processes handle client request separately and do not share the same memory space.	The Servlet technology is based on a threaded architecture.

Table 6.1 – *Continued*.

	scripts is language-dependent.	"Authorizer" role for the applications and helps to restrict accessibility to portions of the program that are purely user-specific.	like network or file-system access, database connectivity, etc. and decides if those actions are permitted. Running the Security Manager and monitoring all activities could increase the latency on handling client requests as there is additional time involved in evaluating security policies for each network or file access.
Database Access	CGI programs open and close a database connection for each new request that requires database access. Thus, these connections are non-persistent.	FastCGI does not support caching of database connections like traditional CGI. The initialization cost for a database connection may be high as a new connection needs to be opened and closed for the lifetime of the request. But for applications that require limited database access this would mean considerable savings on system resources.	Servlets are far more efficient in managing database access. They maintain a pool of database connection objects. Every time there is a request made that requires a database connection, it checks for an available object in this pool and reuses it otherwise a new object is initialized. This object is then returned to the pool once the access is completed. Hence the database connection objects are persistent and may be reused to serve multiple client requests.

The above table summarizes the comparison between the different web technologies based on some of the important features. The Servlet technology is rather new when compared to CGI but is feature rich, is based on a superior architecture that uses multi-threading, and provides the benefits of implementing in Java programming language.

CGI scripting may be implemented in a variety of languages like Perl, PHP, etc. We consider the use of extension APIs to improve the performance of the scripting language on the web server. Traditionally, an external Perl interpreter needed to be loaded into memory each time a web request was made to the server. The interpreter would then parse the CGI script and process the client request. This obviously resulted in a large performance overhead as the web server would need to spawn a new instance of the interpreter for every new request. Extension APIs like mod_perl [MODPERL] help to overcome this overhead and extend the functionality of the web server by incorporating the programming features of Perl. The Perl interpreter is embedded into the web server and is persistent. This means that it does not need to be started with each new request.

The performance comparison between these technologies has been studied and results presented in various papers. Based upon the results presented in [GOUSIOS02] [WANG00], the following conclusions may be made:

- FastCGI outperforms other technologies in the maximum number of concurrent requests that it handles in a given interval of time. Although both FastCGI and Servlets reuse processes/threads to serve client requests, the fact that FastCGI may be running locally on the web server reduces the communication overhead. The web server uses full-duplex pipes to connect to the processes when run locally in comparison to the overhead of using TCP connection which are a factor slower. Using the servlet technology, the application server which provides the servlet container, communicate with the web server over TCP connections. The

cost of establishing a TCP connection and the network latency are factors that affect the rate at which Servlets can handle client requests.

- The average response time (response rate) was measured as a benchmark test in [GOUSIOS02]. FastCGI and mod_perl were found to be faster than Servlets by about a factor of 4. The results indicate the effectiveness of the solution provided by FastCGI as an improvement over traditional CGI. FastCGI performs in-memory caching also described as session affinity. This is configured as part of the web server using FastCGI. The client requests are routed to one of the processes in the pool based on the content requested. Each process is allocated its own cache space which is accessed to process the client request. By effectively utilizing the process cache, the application is able to reduce the amount of disk I/O and the processor utilization. Another advantage seen using both FastCGI and mod_perl is using a multi-threading within each process to handle a large number of client request. Servlets also benefit from using multi-threading in handling client requests.
- The results presented by the authors in [GOUSIOS02] indicate that FastCGI in particular provides superior behavior based on the measurements provided by the performance metrics. These results do not guarantee that a single-solution would clearly out-perform the rest under all load conditions. One of the potential causes for the slowdown on the Servlet implementation is the thread synchronization. Under high load conditions, possibility of resource contention like access to database connections can occur.

We have studied various web technologies that are presently available for web-development. The performance results based on the works done in this area were researched to gain better understanding of the technology and the relative benefits provided by each of them.

Present day grid-portals have embraced different technologies based upon their assessment and individual needs. The ASC portal has based its implementation of the Java Servlets technology with Java Server Pages for front-end GUI development. The solution provided by the WebCom portal does not take into accounts the benefits presented by technologies like FastCGI and mod_perl.

The CSEGrid portal employs the CGI technology to implement the portal service. We have chosen to use the mod_perl extension for Apache Web server to reduce the performance overhead of processing CGI scripts to handle client requests. Our choice was partly influenced by the ability for rapid website development using the CGI technology, feature-rich APIs that provide the functionality required in integrating the portal solution with the back-end resources, among others. Although, traditional CGI does not provide the performance benefits unlike other solutions, use of extension APIs like mod_perl help bridge this gap. FastCGI promises to be one of the solutions that would immensely help in improving the scalability of the portal service in future development plans.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Grid-enabled web portals are emerging as popular mechanisms for providing the user with ubiquitous access to grid computing resources and services through easy-to-use interfaces. The portal hides the complexity of the grid from the user and presents a uniform and transparent access to the grid through the portal interfaces. The convenience and availability of access to grid services anytime and anywhere has drawn considerable interest within the scientific community.

7.1 Conclusions

In this work we have identified the need for a web-based grid portal and described the construction of the CSEGrid portal system. The prototype of the portal system was developed to provide access to some of the core grid services like grid authentication, job submission, resource monitoring, and file transfers across grid-enabled resources through a set of web interfaces. We have studied the security provisions made in other portal implementations and found the need for a credential management service that would require the user to delegate trust to the portal system. Using this service the portal is then able to:

- Provide better protection to the user's credentials against accidental or intentional compromise.
- Ensure the portal services are available to the user at all times without the user having to worry about expiring credentials.

The CSEGrid portal utilizes middleware tools to hide the underlying grid resources and present transparent access to the grid through simple and lightweight interfaces that provides the user with the ability to access grid services from any remote computer that has a web browser installed and is connected to the Internet.

7.2 Future Work

The CSEGrid portal project is an ongoing effort in developing a web-based solution for providing access to grid-based resources available at various sites on campus.

- The preliminary step would be to deploy the prototype of the system on a subset of available grid resources that are currently being used to perform real-world computations.
- The current prototype provides access to core Globus grid services through the portal interfaces. Currently the default job manager "fork" is used to handle the jobs. In this setup, the jobmanager forks a new process to launch a new job and terminates when the job finishes execution. In the high performance computing environment, batch scheduling systems like PBS, LSF, etc. are commonly employed. Jobs which are required to execute in a particular sequence are bundled together and submitted to batch scheduler. The current prototype needs to be

enhanced to support interfaces to enable submission and monitoring of batch jobs that use PBS and others.

- Apart from the ease-of-use and accessibility, performance of the portal system is another feature that both the user and the developer of the portal system are interested in. Although network bandwidth and latency can affect the performance "observed" by the user, it becomes necessary to study the behavior of the portal system under varying load conditions. The web server which acts as a single point of entry for users can become a serious bottleneck with increasing number of users and user activity. Various load balancing schemes for web servers were studied. Incorporating load balancing features into the portal server design would transform into better performance and would the system fault-tolerant as requests from users can be served by any one of the web servers in the cluster. It would be worthwhile investigating into this area to provide better service to the users.
- The trust management service proposed in the developed system performs credential management operations on credentials obtained from the local CA. It has been observed that users generally possess multiple certificates, each issued by a different CA that is used to authenticate the user at various resource sites. Management of multiple credentials is seen not only to be cumbersome but also often resulting in the credentials being lost or stolen due to the user's carelessness. It would be interesting to investigate provision of a credential management service in such a scenario and understand the constraints that apply in providing such a service that would benefit the user.

APPENDIX A

TERMS AND CONCEPTS

Certificate policy (CP): A named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. [RFC2527]

Certification Practice Statement (CPS): A statement of the practices which a certification authority employs in issuing certificates. [RFC2527]

Security Policy (SP): A security policy is a formal statement of the rules by which people who are given access to an organization's technology and information assets must abide. [RFC2196]

Commodity Grid (CoG) Toolkit: A CoG Toolkit defines and implements a set of general components that map Grid functionality into a commodity environment / framework. [LASZEWSKI]

Non-repudiation: Non-repudiation provides assurance that a specific action occurred. Non-repudiation controls prevent an individual from being able to deny receipt, submission, or delivery of a message. [KING01]

REFERENCES

[BLUEPRINT] I. Foster and C. Kesselman, *The GRID: blueprint for a new computing infrastructure*, Morgan Kaufmann Publishers, 1999.

[ASCPORTAL] M. Russell, G. Allen, G. Daues, I. Foster, J. Novotny, E. Seidel, J. Shalf and G. Laszewski. *The Astrophysics Simulation Collaboratory: A Science Portal Enabling Community Software Development*, 10th IEEE International Symposium on High Performance Distributed Computing, August 07-09 2001.

[GENIUS] R. Barbera, A. Falzone and A. Rodolico, *The GENIUS Grid Portal*, Conference for Computing in High Energy and Nuclear Physics, March 24-28 2003.

[WEBCOM] G. He and Z. Xu, *Design and Implementation of a Web-based Computational Grid Portal*, Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03), IEEE Computer Society, October 13-17 2003.

[LEGION] A. Natarajan, A. Nguyen-Tuong, M. Humphrey and A. Grimshaw, *The Legion Grid Portal*, GCE 2001.

[GRIDSPHERE] J. Novotny, M. Russell, O. Wehrens, *GridSphere: A Portal Framework for Building Collaborations*, 1st International Workshop on Middleware for Grid Computing, Rio de Janeiro, June 15 2003.

[MYPROXY] J. Novotny, S. Tuecke, V. Welch. *An Online Credential Repository for the Grid: MyProxy*, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), August 07-09 2001.

[FOSTER98] I. Foster, K. Czajkowski, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, *A Resource Management Architecture for Metacomputing Systems,* Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, 1998.

[GRIDPORT] M. Thomas, S. Mock, M. Dahan, K. Mueller and D. Sutton, *The GridPort Toolkit: a System for Building Grid Portals*, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), August 07-09 2001.

[KEAHEY02] K. Keahey, V. Welch, *Fine-Grain Authorization for Resource Management in the Grid Environment, Proceedings of Grid2002 Workshop*, 2002.

[LDAP] T. Howes, M. Smith and G. Good, *Understanding And Deploying LDAP Directory Services*, First Edition, McMillan Technical Publishing, 1999.

[BELL73] D.E. Bell, I.J. LaPadula, *Secure Computer System: Unified exposition and Mutics interpretation*, MITRE Corporation, MTR-2997, 1976.

[CLARK87] D.D. Clark, D.R. Wilson, *A Comparison of Commercial and Military Security Policies*, Proceedings of the IEEE Symposium on Security and Privacy, CS Press, Los Alamitos, USA, 1987.

[DENN76] D.E. Denning, *A Lattice Model of Secure Information Flow*, Communication of the ACM, 19(5), May 1976, pgs 236-243.

[LASZEWSKI] G. vonLaszewski, I. Foster, J. Gawor, *CoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids*

[KING01] C.M. King, C.E. Dalton, T.E. Osmanoglu, *Security Architecture: Design, Deployment & Operations*, McGraw-Hill Publications, 2001.

[NANDA99] Y. Hu, A. Nanda, Q. Yang, *Measurement, Analysis and Performance Improvement of the Apache Web Server*, 18th IEEE International Performance, Computing, and Communications Conference (IPCCC'99), Phoenix, February 10-12, 1999.

[GOUSIOS02] G. Gousios, D. Spinellis, *A Comparison of Portable Dynamic Web Content Technologies for the Apache Server*, Proceedings of the 3rd International System Administration and Networking Conference Proceedings, Maastricht, May 2002.

[WANG00] A. Wu, H. Wang, D. Wilkins, *Performance Comparison of Alternative Solutions for Web-to-Database Applications*, Proceedings of the Southern Conference on Computing, Mississippi, October 26-28, 2000.

[LEGIONPAGE] http://legion.virginia.edu/index.html

[ALLIANCE] http://www.extreme.indiana.edu/xportlets/project/index.shtml

[NEESGRID] http://www.neesgrid.org/index.php

[NSF-NMI] http://www.nsf-middleware.org

[ENGINFRAME] http://www.enginframe.com/docum/index.html

[SERVLETS] http://java.sun.com/products/servlet/docs.html

[JAVA] http://sun.java.com

[GPDK] http://doesciencegrid.org/projects/GPDK/

[JAVACOG] http://www-unix.globus.org/cog/java/index.php

[HOTPAGE] https://hotpage.npaci.edu/

[GRIDBUS] http://www.gridbus.org

[DIST.NET] http://www.distributed.net

[SETI] http://setiathome.ssl.berkeley.edu/

[GLOBUSPAGE] http://www.globus.org

[OPENLDAP] http://www.openldap.org

[RFC2252] http://www.faqs.org/rfcs/rfc2252.html

[RFC2527] http://www.ietf.org/rfc/rfc2527.txt

[RFC2511] http://www.faqs.org/rfcs/rfc2511.html

[RFC2196] http://www.faqs.org/rfcs/rfc2196.html

[APACHE] http://www.apache.org

[MODSSL] http://www.modssl.org

[STRONGHOLD] http://www.redhat.com/software/stronghold/

[PERL] http://www.perl.org

[OPENSSL] http://www.openssl.org

[BERKELEYDB] http://www.sleepycat.com/download/db/index.shtml

[TCPW]http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-tcpwrappers.html

[RFC2246] http://www.ietf.org/rfc/rfc2246.txt

[FAST-CGI] http://www.fastcgi.com/devkit/doc/fastcgi-whitepaper/fastcgi.htm

[MODPERL] http://perl.apache.org/

BIOGRAPHICAL INFORMATION

Srikant D. Rao is a graduate student in the Computer Science and Engineering Department at UT Arlington. He received his Bachelor of Engineering degree in Electrical Engineering with commendation from Kuvempu University, Shimoga, India in August, 1998.