

Interoperability and Trust Management Solutions for Establishing Secure Grids

A Thesis submitted

for the award of degree of

Doctor of Philosophy

by

Shashi

(Regn. No. 90603001)

Under the Guidance of

Dr. Seema Bawa

Professor, Computer Science and Engineering Department

Dean of Student Affairs



**Computer Science and Engineering Department
Thapar University, Patiala -147004, INDIA**

June 2011

To my Parents . . .

Contents

List of Figures	xi
List of Tables	xii
Certificate	xiii
Acknowledgement	xiv
Abstract	xvi
1 Introduction	1
1.1 Grid Computing: An Overview	2
1.1.1 Evolution of Grid Computing	2
1.1.2 Grid Components	7
1.1.3 Grid Architecture	9
1.1.4 Grid Standards	12
1.2 Grid Security: The Research Motivation	14
1.2.1 Grid Security Requirements	14
1.2.2 Grid Security Issues	19
1.2.3 Challenges in Grid Security	24
1.3 Limitation with existing approaches	29
1.4 Thesis Contributions	29
1.5 Thesis Organization	31

2	Literature Review	34
2.1	Grid Security Technologies and Standards	34
2.1.1	Grid Security Infrastructure (GSI)	35
2.1.2	Message-level security	35
2.1.3	WS-Security	36
2.1.4	Security Assertion Markup Language (SAML)	37
2.1.5	Extensible Access Control Markup Language (XACML)	38
2.2	Grid Security Challenges	40
2.2.1	Integration	41
2.2.2	Interoperability	43
2.2.3	Trust Management	53
2.3	Interoperability and Trust Management Issues	76
2.3.1	Authentication	76
2.3.2	Authorization	79
2.4	Security in Grid Middleware	84
2.4.1	GLOBUS TOOLKIT	86
2.4.2	UNICORE	90
2.4.3	Condor	92
2.4.4	SUN Grid Engine	94
2.4.5	LEGION	96
2.5	Problem Formulation	102
2.6	Objectives of the Thesis	103
3	The Proposed Interoperability Solutions: TUX-INTERO, TUX- INTEROP and PIE	106
3.1	Interoperability	107

3.2	TUX-INTERO: A Web Services based interoperability solution	107
3.2.1	Taxonomy of TUX-INTERO	108
3.2.2	Architecture of TUX-INTERO	110
3.2.3	Implementation details of TUX-INTERO	117
3.3	TUX-INTEROP: A Portal for secure interoperation of grids	127
3.3.1	Taxonomy of TUX-INTEROP	128
3.3.2	Architecture of TUX-INTEROP	129
3.3.3	Implementation details of TUX-INTEROP	133
3.4	PIE: A Grid Portal for Interoperability	137
3.4.1	Taxonomy of PIE	137
3.4.2	Architecture of PIE	137
3.4.3	Implementation details of PIE	141
3.5	Conclusions	146
4	The Proposed Trust Management Solution: TUX-TMS	147
4.1	Trust Management	147
4.2	TUX-TMS: A flexible reputation based trust management solution . . .	149
4.2.1	Taxonomy of TUX-TMS	149
4.2.2	Trust Evaluation Parameters	150
4.2.3	Trust Parameters	151
4.2.4	Partitioning of the Users	158
4.2.5	Architecture of TUX-TMS	161
4.2.6	Implementation details of TUX-TMS	175
4.3	Conclusions	188
5	Deployment, Testing and Validation of the Proposed Solutions	191

5.1	TUX-CITMS: Thapar University Extensible-Collaborative Interoperability and Trust Management Solution: A Grid Testbed	192
5.1.1	Evaluation Parameters	195
5.1.2	Experimental Details	197
5.1.3	Deploying TUX-INTERO for providing interoperability across multiple platforms at Thapar University Campus	199
5.1.4	Deploying TUX-TMS for establishing and evaluating Trust across Thapar University	205
5.1.5	Testing Details	212
5.1.6	Discussion	242
5.2	TUX-INTEROP: A Portal for secure interoperation of grids	245
5.2.1	Evaluation Parameters	246
5.2.2	Experimental Details	247
5.2.3	Testing Details	248
5.2.4	Discussion	255
5.3	PIE	257
5.3.1	Evaluation Parameters	257
5.3.2	Experimental Details	258
5.3.3	Testing Details	259
5.3.4	Discussion	268
5.4	Conclusions	269
6	Conclusions and Future Directions	271
6.1	Summary of Contributions	271
6.2	Future Directions	273

References **275**

 List of Publications 295

 Recognition for Work done 297

List of Figures

1.1	Evolution of Grid Computing	4
1.2	Layered Architecture of Grid Computing	12
1.3	Grid Security Issues	20
2.1	Web Services Security Specifications	37
2.2	Components of Grid Security Model	39
2.3	HPC Grid Interoperability Demonstration	49
2.4	Grid Security Infrastructure	90
2.5	Unicore Architecture	91
2.6	Legion Architecture	97
3.1	High Level Architecture of TUX-INTERO	111
3.2	Detailed Architecture of TUX-INTERO	112
3.3	Class Diagram of Grid entities in TUX-INTERO	113
3.4	Resource Manager in TUX-INTERO	114
3.5	High level Architecture of TUX-INTEROP User Interface	115
3.6	Deployment of TUX-INTERO	116
3.7	Desktop Application designed in .Net for invoking ASP.Net Web Services in TUX-INTERO	118
3.8	Invoking Web Services	119

3.9	Invoking ASP. NET Web Services Using PHP	120
3.10	Extensible Stylesheet Language Transformations (XSLT)	121
3.11	Architecture of TUX-INTEROP Portal	131
3.12	Resource Description in TUX-INTEROP	134
3.13	Architecture of PIE	138
3.14	Domain.xml of PIE	141
4.1	Phylogenetic tree of Users in TUX-TMS	160
4.2	Architecture of TUX-TMS	161
4.3	Use Case Diagram for User Authentication in TUX-TMS	163
4.4	Activity Diagram for User Authentication in TUX-TMS	164
4.5	Activity Diagram for New User Registration in TUX-TMS	165
4.6	An activity diagram of TUX-TMS	177
4.7	Use case of Service Requester in TUX-TMS	179
4.8	Use case of Service Provider in TUX-TMS	179
4.9	Use case of Administrator in TUX-TMS	180
4.10	Activity Diagram for Administrator in TUX-TMS	181
4.11	Stored Procedures in TUX-DB to query and update values	186
4.12	Functions in TUX-DB to query and update values	187
4.13	Tables in TUX-DB to store values	188
5.1	Integration of TUX-INTERO and TUX-TMS	193
5.2	TUX-CITMS: Grid Testbed	193
5.3	User Records of TUX-CITMS	196
5.4	Services exposed to Service Requester in TUX-CITMS	197
5.5	Administrator Screen in TUX-CITMS to View Users	200

5.6	Administrator Screen in TUX-CITMS to Add/Modify Users	201
5.7	Services Available to the Service Requester	202
5.8	Desktop Application 1 of TUX-INTERO	203
5.9	Desktop Application 2 of TUX-INTERO	204
5.10	Invoking ASP. NET Web Services Using PHP	205
5.11	Web Services exposed to TUX-INTERO Desktop Application	206
5.12	Service Records of the Services Available in TUX-CITMS	207
5.13	Invoking Web Services from TUX-INTERO Desktop Application	208
5.14	Result showing success of Test Case 1: Interoperability between nodes hosted in different versions of same Operating System	209
5.15	Result showing success of Test Case 2: Interoperability between nodes hosted on different Operating Systems	210
5.16	Requesting a Service by a Service Requester for accessing	211
5.17	Result showing success of Test Case 12: View Trustworthiness Scores (Service Provider)	212
5.18	Accessing a Computational Service by a Service Requester	213
5.19	Submitting Feedback for the Services Accessed by Service Requester	214
5.20	Feedback Form of Service Requester	215
5.21	Closing a Service by Service Requester to use again	216
5.22	Feedback Form of Service Provider	217
5.23	Registering a Service by a Service Provider in TUX-CITMS	218
5.24	Service Provider Screen showing the services used by Service Requester	219
5.25	Result showing success of Test Case 3: Interoperability between systems hosted in different Operating Systems and Middleware	220
5.26	Trust Inheritance and Risk Assessment (Service Provider)	224

5.27 Trust Inheritance and Risk Assessment (Service Requester)	224
5.28 Trustworthiness and Reputation Score (Service Requester)	226
5.29 Trustworthiness and Reputation Score (Service Provider)	228
5.30 Database results for Service Provider	230
5.31 Database results for Service Requester	231
5.32 Result showing success of Test Case 15: Number of transactions were increased after deploying TUX-TMS into Grid Computing Environments	238
5.33 Result showing success of Test Case 16: Number of transactions were increased and threats decreased during after deploying TUX-TMS into Grid Computing Environments	239
5.34 Result showing success of Test Case 1:TUX-INTEROP Login Screen designed in Gridsphere	250
5.35 TUX-INTEROP Portlets	251
5.36 Result showing success of Test Case 2: TUX-INTEROP: File Manage- ment Portlet of TUX-INTEROP	252
5.37 Credential Management Portlet of TUX-INTEROP	253
5.38 Result showing success of Test Case 3: Resources in TUX-INTEROP .	254
5.39 Result showing success of Test Case 4: Job Management Portlet of TUX- INTEROP	255
5.40 Monitoring a server in TUX-INTEROP	256
5.41 Certificate Manager	259
5.42 Mapping Roles to the PIE users	260
5.43 Login Screen of PIE	261
5.44 Result showing success of Test Case 1: Credential Management Portlet of PIE	262

5.45	Result showing success of Test Case 4: Registry Management Portlet of PIE	263
5.46	Result showing success of Test Case 3: File Management Portlet of PIE	264
5.47	Result showing success of Test Case 2: Job Management Portlet of PIE	264
5.48	Writing a job using Job Service Description Language (JSDL)	265
5.49	Result showing success of Test Case 5: Creating and embedding new portlets using a Project	266

List of Tables

2.1	Trust Level	70
2.2	Responses to Comparison in Tables 2.3 and 2.4	72
2.3	Trust Management Systems in various Computing Paradigms	73
2.4	Trust Management Systems in various Computing Paradigms (continued)	74
2.5	Comparison of Middleware and Projects in terms of Security	85
2.6	Security in Grid Middleware	99
3.1	Comparing existing Interoperability solutions with Proposed solutions .	145
4.1	Trust Level	155
4.2	Partitioning the users	159
4.3	Risk Probabilities and Impacts of Risk	185
4.4	Database Tables	189
4.5	Comparing existing Trust Management solutions with Proposed solution	190
5.1	Grid Setup	199
5.2	Security Threats on TUX-CITMS	241

Certificate

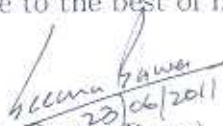
I hereby certify that the work which is being presented in this thesis entitled "Interoperability and Trust Management Solutions for establishing Secure Grids," in full fulfillment of the requirement for the award of degree of "Doctor of Philosophy" submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Seema Bawa and refers other research works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.


(Shashi)

Regn. No. 90603001

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.


(Dr. Seema Bawa)

Professor,

Department of Computer Science Engineering

Dean of Student Affairs

Thapar University

Patiala, 147004

Punjab, INDIA.

Acknowledgements

I pay my reverence and gratitude to the Almighty for his everlasting blessings. This thesis arose out of years of research that has been done since I joined Thapar University. I have known a great number of people during these many years, whose contribution in assorted ways to the research and the making of the thesis deserved special mention. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

Firstly, I offer my gratitude to my esteemed supervisor **Dr. Seema Bawa**, Professor, Thapar University, Patiala, for her supervision, advice and guidance from the very early stage of this research as well as giving me extraordinary experiences throughout the work. Above all and the most needed, she provided me unflinching encouragement and support in various ways. Her truly scientific intuition has made her as a constant oasis of ideas and passions in science, which exceptionally inspire and enrich my growth as a student, a researcher and a scientist want to be. I am indebted to her insightful comments and for putting so much energy for this work and transforming me as an individual. Her experience, strength, tenderness and willfulness, has taught me the lessons of life, which are of immense help to me to take decisions in my every endeavor. I am profoundly obliged to **Dr. Maninder Singh**, Associate Professor and Head, CSED, for his whole-hearted support and motivation.

I take this opportunity to thank **Dr. Abhijit Mukherjee**, Director, Thapar University, for all the facilities which have been immensely helpful for the completion of my work. I gratefully acknowledge the financial support of Thapar University extended as a Teaching Assistant in Computer Science and Engineering Department, throughout my work.

I am thankful to my Doctoral committee members for their constructive comments and regularly ensuring the progress of my research work.

I am thankful to all the **faculty** members of CSED and my labmates **Dr. Sarbjeet Singh, UIET, Seemu, Nidhi, Ratinder** and **Rajni** at Center of Excellence in Grid Computing for their support. I am also thankful to the office and laboratory staff of Computer Science and Engineering Department for all the cooperation and help.

Uncountable words of thanks to **Mr. Sanjay Verma**, TCS who helped me in designing and developing the research idea into a full fledged system and gave me his precious time and guidance.

I also acknowledge the co-operation and encouragement extended to me by my fellow Research scholars and my good friends **Meenakshi Malik** and **Zinki Jindal** at Thapar University. Research had been enjoyable experience in their company. I would also like to thank my friends **Neetu, Gurpreet, Navdeep Dhami, Henna, Jyotsana, Vandana Gudwani, Arsh, Aastha** and **Dr. Hitashi Lomash** who have been very kind and supportive. I am also thankful to **Dr. Amita** and **Dr. Puneet Sharma** who have been family like support to me. Last but not the least, I would express my gratitude to **Mrs. Aruna** and **Mr. Rajesh Dhankhar** who have been standing by me through all the thick times and in my every endeavor.

I offer my deepest gratitude to my parents, **Smt. Kamla Bhanwar** and **Sh. D.P. Bhanwar**, my sister, **Miss Seema** and my brothers, **Mr. Sandeep** and **Mr. Sunny**, for their love, encouragement and motivation and for their confidence in me.

Finally, I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

Patiala

June, 2011

Shashi

Abstract

Security remains one of the fundamental challenges for adoption of Grid computing in a wider commercial context. Security challenges include finding suitable resources across the domains, irrespective of underlying platforms, leading to Interoperability challenges. Also in Grid computing scenario traditional assumptions for establishing and evaluating trust, do not hold good, there is an inherent need to establish trust between unknown players (service requesters, service providers, etc.). This leads to another type of security challenges, trust management challenges.

The focus of this thesis is to address Interoperability and Trust Management challenges in Grids and provide solutions for these. As an outcome, three Interoperability Solutions: Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE) have been proposed and developed. As Trust Management solution: Thapar University Extensible Trust management System (TUX-TMS) has been proposed and developed.

TUX-INTERO is a Web Services based interoperability solution which discovers and integrates resources based on various platforms and middleware. TUX-INTEROP is a grid portal for secure interoperation of grids. PIE is grid portal which integrates resources from various middleware and allows access to end users at a single interface. TUX-TMS is a flexible Reputation based Trust Management system which combines the merit of policy proof and reputation evaluation.

TUX-INTERO and TUX-TMS have been integrated, deployed and tested on Thapar University Extensible-Collaborative Interoperability and Trust Management Solution (TUX-CITMS)- a Grid Testbed. This testbed spans across whole Thapar University Campus. Testing results have been presented in the form of numerous test cases. TUX-

INTEROP and PIE have been deployed and tested further on Thapar University (TU) Campus Grid using a set of test cases. Every test case defines a testing criteria, test inputs, expected outputs and finally validates the outcome to be correct/ as expected. The results successfully exhibit interoperability between resources on multiple platforms and middleware. The testing results collectively demonstrate that the proposed solutions are working and can be effectively used to address interoperability and trust management challenges to establish secure Grids.

Chapter 1

Introduction

The term ‘Grid’ was coined in mid 1990s to describe a large scale distributed computing infrastructure for advanced science and engineering applications [1], [2]. Since then, it has emerged as an important and interesting field that is different from conventional distributed computing by having its focus on large scale resource sharing, innovative applications and in some cases, high performance orientation [3], [4], [5]. Grid computing has revolutionized the way scientific discoveries and research is done in the world [6], [7]. A wide number of projects have been built around it [8]. This popular distributed computing paradigm has also signified the power of curiosity driven exploration of unknown facets of science [9], [10], [11]. It has become industry slogan today due to its unique features. Although a Grid was envisioned to be a decentralized system spanning multiple administrative domains, encompassing multiple platforms and systems, that provides non trivial qualities of service to a broad audience of clients, where both the set of users and resources can vary dynamically [12], [13]. But, the threat of compromising this massive infrastructure to the hands of malicious users have confined it to be used only in secure environments under strict administrative control. The term “Grid” has been inspired by the analogy to power Grids, which give access

to electricity as a utility to the end users on per-unit-charge basis, irrespective of its origin and source of generation. Grid is envisioned to be available as a utility service to end users, on nominal charges at their doorstep. The fundamental objective of Grid Computing is to unify distributed computing resources independent of scale, hardware and software in order to achieve processing power in unprecedented ways.

This chapter briefly discusses evolution of Grid Computing, its characteristics, components and architecture. Grid security requirements, issues and challenges, which are the focus of this research work are also addressed in this chapter. The organization of the thesis is presented at the end of the chapter.

1.1 Grid Computing: An Overview

Grid extends the existing distributed computing resources further into a more unified and collaborative structure. Grid also enables the heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of virtual resources [14]. The next section explores evolution of Grid through three different generations.

1.1.1 Evolution of Grid Computing

The notion of Grid Computing had already been explored in very early days of computing (as shown in Figure 1.1) leading to the evolution of Grid through three different generations [15], as explained below:

1.1.1.1 First Generation: Early Metacomputing Environments

In the early 1990s, the first generation efforts were marked by the emergence of metacomputing projects, which aimed at linking supercomputing sites to provide access to an end user to massive computational resources. Two representative projects of first generation; Factoring via Network-Enabled Recursion (FAFNER) [16] and The Information Wide Area Year (I-WAY) [17], are considered as the pioneers of Grid Computing. FAFNER was created through a consortium, to factor RSA-130¹ using a numerical technique called Number Field Sieve. I-WAY was an experimental high performance network that connected several high performance computers spread over seventeen universities and research centers using mainly Asynchronous Transfer Mode (ATM) technology. FAFNER was the precursor of initial Grid projects such as The Search for Extraterrestrial Intelligence at Home (SETI@home) [18] and Distributed.Net [19]. I-WAY was the predecessor of the Grid middleware projects such as Globus [20] and Legion [21].

1.1.1.2 Second generation: Core Grid Technologies

The I-WAY project paved the path for the evolution of the Grid to the second generation, which focused on the development of middleware to support large scale data access and computation. Grid middleware is designed to hide the heterogeneous nature of resources in order to provide users and applications with a homogeneous and seamless environment. It involved the development of core Grid technologies: Grid Resource

¹The Rivest, Shamir and Adelman (RSA) public Key encryption algorithm is widely used for enhancing security. The security of RSA itself is based on premise that it is difficult to factor extremely large numbers, in particular, those with hundred numbers. To keep abreast of the state of art of factoring, RSA Data Security Inc. initiated the RSA factoring challenge in March 1991. The Factoring Challenge provides a test bed for factoring implementations and provides one of the largest collections of factoring results from different experts worldwide. FAFNER was setup to factor RSA 130 using a numerical technique called the Number Field Sieve.

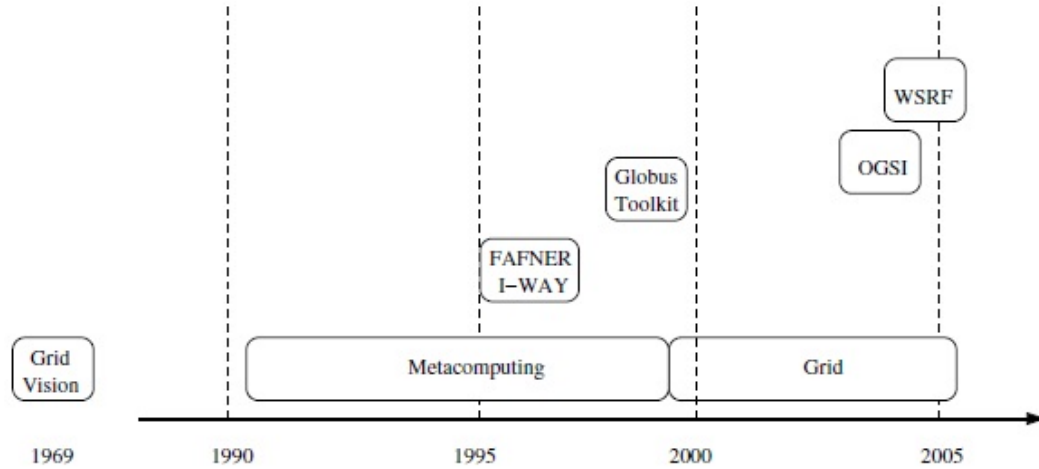


Figure 1.1: Evolution of Grid Computing [13]

Management such as Globus [20], Legion [21], resource brokers and schedulers such as Condor [22], PBS [23], Grid portals such as GridSphere [24] and complete integrated systems such as UNICORE [25], Cactus [26].

1.1.1.3 Third generation

The core middleware for the Grid developed in the second generation provides the basic interoperability to enable large-scale resource sharing. The third generation focuses on the adoption of service oriented model in development of Grid applications. In early 1990s, Grid researchers came together to expand Global Grid Forum (GGF) [27], a community constituting people from industry and academia responsible for standardizing Grid. The Global Grid Forum defined Open Grid Services Architecture (OGSA) [28], which is based on Web services protocols. OGSA was then combined with Grid protocols to define the Open Grid Service Infrastructure (OGSI), which provides a uniform architecture for the development and deployment of Grids and Grid applications. The creation of Web Services Resource Framework (WSRF) [29] completed the convergence

between Web and Grid Service Architecture. These standards have been discussed in detail in the next section. In 2006, Global Grid Forum merged with Enterprise Grid Alliance to form Open Grid Forum (OGF) [27]. Presently, OGF is dedicated to enable business value and scientific discovery by developing open standards for wider Grid adoption. However, for widespread adoption of Grid Computing as an automatic choice in enterprises depends upon the ability of the researchers and practitioners in reducing the pitfalls that lie on the way. Some major pitfalls as identified by OGF are Grid Security challenges such as Integration, Interoperability and Trust Management, which are also the focus of the research work undertaken.

Grid computing is all about achieving greater performance and throughput by pooling resources on a local, national and international level. The latest advances in Grid Computing have enabled the creation of many national and international Grids such as TeraGrid [30] and Open Science Grid [31] in USA, Enabling Grids for E-science (EGEE) [32] in Europe, E-Science [33] Grid in the UK, K*Grid [34] in Korea, NAREGI [35] in Japan, Garuda [36] in India, Our Grid [37] in Brazil, Grid'5000 [38] and Auver Grid [39] in France and DAS [40] in the Netherlands.

A Grid has following characteristics:

Multiple administrative domains and autonomy: Grid resources are geographically distributed across multiple administrative domains and owned by different organizations. The autonomy of resource owners needs to be honored along with their local resource management and usage policies.

Heterogeneity: A Grid involves multiple resources that are heterogeneous in nature and encompass a vast range of technologies.

Scalability: A Grid might grow from a few integrated resources to millions of resources of varied range. Consequently, applications that require a large number of

geographically located resources must be designed to latency and bandwidth tolerant.

Dynamicity or Adaptability: In a Grid, resource failure is a rule rather than an exception. In fact, with so many loosely connected resources in a Grid, the probability of some resource failing to deliver its service is high. Resource managers or applications must tailor their behavior dynamically and use remaining available resources and services efficiently and effectively.

The concept of Virtual Organization (VO) [3] is central to Grid computing. A VO is a dynamic grouping of individuals, multiple groups, physical organizations or administrative domains having their own set of usage policies that it would like to enforce on its resources and services. The resources that can be shared in these various types of Grid are:

Computational resources: The most common resource utilized in a Grid is the processing power provided by the nodes, which are part of the Grid. The processors can vary in speed, architecture and software platforms.

Data storage: Secondary storage in a Grid can be used in interesting ways to increase capacity, performance, sharing and reliability of data. Many Grid systems use mountable networked file systems, such as Andrew File System (AFS), Network File System (NFS), Distributed File System (DFS) or General Parallel File System (GPFS). These filesystems offer varying degrees of performance, security features and reliability features [41], [42].

Communication: The rapid growth in communication speed among disparate nodes today makes Grid Computing practical, compared to the limited bandwidth available when distributed computing was emerging.

Scientific Instruments: Instruments such as telescopes, satellites and sensors etc. can be used for varied purposes in a Grid.

Software and licenses: The Grid may have software installed that may be too expensive to install on every Grid node. Using Grid, the jobs in need of this software are sent to particular nodes in which this software happens to be installed. When the licensing fee of the software is significant, this kind of approach can save considerable amount of expenses for an organization.

Grid Computing handles the task of collaboration of these heterogeneous resources through various components which performs their specified tasks as discussed below.

1.1.2 Grid Components

The Grid is made up of a number of components which interact with each other for enabling successful execution of end user applications. This section unveils various Grid components:

1.1.2.1 Portal/User Interface

Effectively accessing information on the Grid is fairly important and the user interface component which is also known as portal, handles this task for the user. P-GRADE [43], ActiveSheets [44], GENIUS [45], GridScape [46], G-Monitor [47], Gridsphere [24] are some of the existing Grid Portals.

1.1.2.2 Security

Resources on a Grid are expensive, networked, run various confidential applications and handle sensitive or extremely valuable data; making security component, most crucial and of critical concern to be managed and maintained. Grid Security Infrastructure (GSI) [3] is considered the most significant Grid security initiative.

1.1.2.3 Broker

A broker is a master scheduler. It allows a user or a process to request resources for a job. It may perform load balancing across multiple systems. Brokers discover resources for a job, select appropriate nodes and submit the job. There are metaschedulers, superschedulers and simple brokers. The metaschedulers take care of load balancing. Superschedulers discover the resources, select them and submit jobs to them. Basic brokers quote resources or discover resources and distribute data and computations based on the determined cost model [12]. NIMROD-G [48], Gridbus Broker [49], JSS-RB [50], eNANOS [51], GRUBER [52] etc. are some of the brokers available in Grids.

1.1.2.4 Scheduler

A scheduler is needed to locate the nodes on which to run an application is to be run and required jobs are assigned. This can be as simple as taking the next available resource, but often this task involves prioritizing job queues, managing the load and finding idle machines. Once the nodes have been identified, the next logical step is to schedule the individual jobs to run on them. If a set of stand-alone jobs are to be executed with no interdependencies, then a specialized scheduler may not be required. However, if a specific resource is to be reserved which requires different jobs within the application to run concurrently (for instance, if they require inter-process communication), then a job scheduler should be used to coordinate the execution of the jobs. Some of the schedulers available in Grid are CONDOR-G [53], NetSolve [54], GridWay [55] etc.

1.1.2.5 Data Management

If any data must be moved or made accessible to the nodes where an application's job will execute, then there needs to be a secure and reliable method for moving files and

data to various nodes within the Grid. The Globus Toolkit contains a data management component that provides such services. This component is known as Grid Access to Secondary Storage (GASS) [12], which includes facilities such as GridFTP. GridFTP is built on top of the standard File Transfer Protocol (FTP), but adds additional functions and utilizes the Grid Security Infrastructure (GSI) for user authentication and authorization. Therefore, once a user has an authenticated proxy certificate, he can use the GridFTP facility to move files without having to go through a log-in process to every node involved. This facility provides third-party file transfer so that one node can initiate a file transfer between two other nodes [12].

1.1.2.6 Job and Resource Management

To handle core tasks such as launching jobs with specific resources, monitoring the status of those jobs and retrieving results, a job and resource management facility is necessary. CONDOR [22], SUN Grid Engine [56] are few examples of Local Resource Management Systems in a Grid.

Grid components work in harmony for successful execution of end user applications. Furthermore, the establishment, management and exploitation of dynamic, cross-organizational VO sharing relationships require Grid architecture. The next section covers Grid architecture in which protocols, services, application programming interfaces and software development kits are categorized according to their roles in enabling resource sharing.

1.1.3 Grid Architecture

Grid's architecture is often described in terms of "layers", where each layer has a specific function. The higher layers are generally user-centric, whereas the lower layers

are more hardware-centric [57]. Grid architecture can be broken into five layers as mentioned here and also shown in Figure 1.2:

1.1.3.1 Fabric Layer

The first and lowest layer is the Fabric layer which consists of clusters of computing nodes, infrastructure elements such as information servers, master nodes (which coordinate computing node clusters for batch and interactive services), storage servers such as disk servers (e.g. NFS, HTTP, GridFTP), tape infrastructure (tape servers and robotics), network infrastructure and miscellaneous servers (time servers, password and credential servers) [57].

Connectivity Layer: The second layer is the Connectivity layer which defines the core communication such as Internet Protocol (IP), Domain Network Service (DNS), routing and authentication protocols required for Grid specific networking services transactions.

1.1.3.2 Resource Layer

The third layer is the Resource layer which defines protocols such as GridFTP protocol (FTP extensions), Grid Resource Allocation Management (GRAM), Grid Resource Information Service (GRIS) for the secure initiation, monitoring and control of sharing operations on individual resources. The resource layer protocols call fabric layer function to access and control local resources. The resource layer protocols are concerned with individual resources and ignore issues of global state and atomic actions across distributed actions.

1.1.3.3 Collective Layer

The fourth layer is the Collective layer which is responsible for coordinating multiple resources. It contains protocol and services concerned with the global state and captures interactions across collections of resources. They implement a wide variety of sharing behaviors without placing new requirements on the resources. Index servers, Resource brokers (e.g., Condor Matchmaker), Replica catalogs, Replication services, co-reservation, co-allocation services and Workflow management services are all part of the Collective layer.

1.1.3.4 Application Layer

The fifth and topmost layer is the Application layer which comprises of user applications that operate within the Grid environment. Applications are constructed in terms of and by calling upon, services defined at any layer. At each layer in a Grid architecture, there exists:

- (a) Certain protocols that provide access to some useful services such as resource management, data access, resource discovery etc.
- (b) Application Program Interfaces (APIs) whose implementation exchange protocol interacts by passing messages to appropriate service(s), to perform desired actions.
- (c) APIs implemented by Software Development Kits (SDKs) which allows exchange of protocol messages with the appropriate services at different layers.

In the Grid architecture, applications are able to call services defined at any lower layers. Useful services such as resource discovery, data access, resource scheduling and so on are normally not directly mapped to a protocol at a specific layer, but may combine protocol operations from more than one layer. The main focus of the

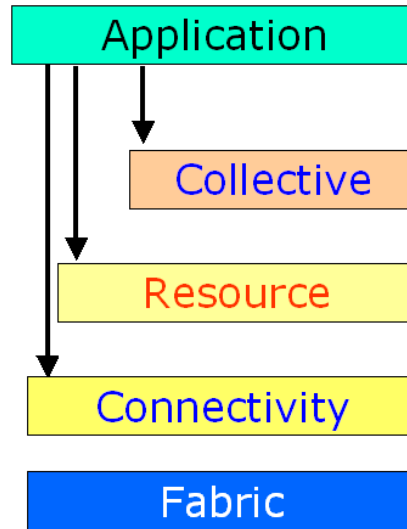


Figure 1.2: Layered Architecture of Grid Computing [57]

architecture is on interoperability among resource providers and users in order to share relationships.

Grid architecture has established requirements for the protocols and APIs that enable sharing of resources, services and code. However, Grids constructed with these different protocols are not interoperable and cannot share essential services at least not without gateways. Therefore, the long-term success of Grid computing requires selecting and achieving widespread development of one set of protocols at the Connectivity layer, which enable different organizations to interoperate and exchange or share resources. However, there should be certain standards available for adopting Grid Computing as a natural choice in terms of collaborative and large scale resource sharing. The standards proposed for Grid Computing are covered in next section.

1.1.4 Grid Standards

The key to realization of Grid vision is standardization. The Global Grid Forum (GGF), now emerged as Open Grid Forum (OGF), is an open community committed

to drive the rapid evolution and adoption of applied distributed computing [27]. OGF accomplishes its work through open forums that build the community, explore trends, share best practices and consolidate the best practices into standards. Two main specifications as proposed by GGF are:

- (i) Open Grid Services Architecture (OGSA)
- (ii) Open Grid Services Infrastructure (OGSI)

The Open Grid Services Architecture (OGSA) aims to define a common, standard and open architecture for Grid based applications. The goal of OGSA is to standardize practically all the services that one commonly finds in a Grid application by specifying a set of standard interfaces for these services [58]. OGSA enables the creation, maintenance and integration of Grid services by virtual organizations. A Grid service provides the controlled management of the distributed and often long lived state that is commonly required in sophisticated distributed applications. OGSA is based on several other Web Services technologies like WSDL and SOAP etc. but aims to be largely agnostic in relation to the transport-level hiding of data.

Open Grid Services Infrastructure (OGSI) [28] is a core component of OGSA which provides a uniform way to describe Grid services and defines a common pattern of behavior for all Grid services. More specifically, the OGSI defines mechanisms for creating, managing and exchanging information among Grid services.

After having an in depth discussion on Grid Computing basics, the next section focusses on Grid security, which is the motivation for this research work.

1.2 Grid Security: The Research Motivation

Grid could have better impact on big commercial and research problems but these problems are not even identified because of absence and lack of understanding of security issues in Grid - so more Outreach is needed. Although, Grid Computing has progressed considerably since its inception, yet, there are numerous challenges, which need to be addressed before realizing the full potential of the Grid. Security is one of the important challenges which needs to be addressed, primarily. Grid Computing focusses on large scale resource sharing of geographically dispersed, diverse and heterogeneous resources located in various administrative domains. These diverse Grid characteristics pose various security requirements, issues and challenges which are of immediate concern to be addressed and are discussed in the subsequent sections.

1.2.1 Grid Security Requirements

Security is a prime concern and necessity of every stakeholder (such as Virtual Organization and the end-user, who can be a Service Provider or Service Requester) in a Grid. Provision of appropriate security within a Grid environment seems to be more challenging than most conventional distributed systems [59]. In a broader sense, any system's security goal is to prevent users who don't have authentic privileges from accessing resources and information. A formal list of Grid security requirements [60], [61], [62], [63], [64] [65], [66], [67], [68], deemed to be essential for supporting scalable, dynamic and distributed Grid environment is presented here:

Entity Authentication: In a Grid environment, there are several types of entities that need to be authenticated. The most common are individual users or service requesters who utilize Grid resources and hosts or service providers, which provide resources and services. Authentication deals with verifying the identity of the end user.

It ensures that end user actually is the identity which he/she is claiming to be. As multiple authentication mechanisms exist, Grid environment must provide integration points for them and the means for conveying the specific mechanisms utilized in any given authentication operation.

Single Sign-on: Participants in a Grid environment often need to co-ordinate and communicate with multiple resources to accomplish a single job. It may be overly burdensome to the participants if they have to manually authenticate with each resource, for example, by typing in a passphrase. Therefore, there should be mechanisms which allows user to perform an authentication process only once to the first resource (when a job is initiated) and do not have to perform any further authentication with any other resource.

Delegation: Grid environment must provide facilities for delegation of access rights from requesters to the services. It is a common occurrence in the real world for a person or organization to delegate authority to an agent: another person or organization who will act on their behalf. In the context of distributed computing systems such as Grids, users will often need to delegate their authority to a service which will act on the user's behalf to carry out some work. In turn, this service may need to delegate authority further to complete the task.

Identity Mapping: In general, authorization can be performed by checking a given identity against a list of allowed identities. In some cases it may be necessary to map an identity from one form to another. An approach used for Grid authorization is to map from the identity presented in a client certificate to a local system account. This is appropriate where resources are protected by the local operating system. The identity can be mapped to an individual account, which the administrator of the system could use to uniquely identify the user, or it could map into a pool of accounts shared among

the users belonging to a virtual organization or some other grouping. Alternatively, accounts can be generated automatically for authorized users.

Credential Life-span and Renewal: The credential must be limited to a reasonable lifetime, to limit the risk of compromising a user's credential especially when the user is performing or using single sign-on and delegation services. In many cases, it may be difficult to predict accurately the credential lifetime required for a specific job. In those cases, where the job takes a longer execution time than the lifetime of the issued credential, the user needs to be notified before the credential expires so that it can be renewed on time, by the user.

Confidentiality: Confidentiality indicates that all data sent by users should be accessible to only legitimate receivers. It deals with the confidentiality of the messages or information that flow over the transport mechanisms. The confidentiality requirement includes point-to-point transport as well as store-and-forward mechanisms.

Message Integrity: Message Integrity indicates that all data received should only be sent/modified by legitimate senders. Message Integrity requirement provides mechanisms to detect unauthorized changes to messages. The use of message or document level integrity checking is determined by the policy, which is tied to the offered quality of service.

Authorization and Access Control: Grid applications require access to resources, which may be located in different organizational domains with different owners. The access rights of a user should be based on authorization policies defined by his virtual organization, the resource owners and local system administrators. This process is usually complementary to entity authentication, so that policy enforcement can be targeted at a specific user. Authorization is concerned with granting an entity rights or permissions to a resource. In discretionary access control (DAC) systems each resource

(a file, for example) has an owner who is responsible for permitting or denying access to the resource. An example is the typical Unix file permissions. Each file has an owner and the owner can decide if it should be readable by no-one else, by a chosen group, or by everyone. Mandatory access control (MAC) systems impose access restrictions on their users and the permissions they can grant. Resources and users have security levels and a user's clearance level must be compatible with that of the resource to gain access. A system policy can enforce that user is not permitted to lower the security level of a resource. These forms of access control centre on the user. Role-based access control (RBAC) grants permissions to particular roles and any user who assumes a particular role will have those permissions. Attribute-based access control (ABAC) extends this to any attribute a user might have. While roles and attributes for an operating system can be assigned to users by a local administrator, for distributed systems a network service may be provided that can issue credentials representing a user's roles, attributes or permissions. In DAC-type systems, the policy governing authorization is usually closely tied to the resource: for example, the permissions stored in the Unix file-system. However, it is often desirable to manage policies independently of the resources, particularly when using an RBAC or ABAC model of access control. In such a system, the authorization decision to permit or deny access may be made separately from the enforcement of the authorization. For example, a user wishing to access a resource contacts a service. The service contains a policy enforcement point (PEP), which contacts a policy decision point (PDP). The PDP evaluates the user's identity, attributes, etc. against an authorization policy and returns a decision to the PEP. The PEP enforces the decision and permits or denies access.

Integration and Interoperability: Each organizational domain has its own security infrastructure which supports a set of security mechanisms. These may be different

from other domains or hosting environments. Hence, a practical Grid security architecture must be able to support and integrate with current security mechanisms. Similarly, adoption of new standards or technologies should place high emphasis on solving potential integration issues that may arise. In addition, interoperability between communicating parties should be achievable at various levels such as transport-level, message-level and service-level.

Trust: Trust requirement deals with trust related aspects like specifying trusted credentials, trust policies and determining trustworthiness of target service. Security architecture must allow both, the service requester and the service provider to specify and enforce trust requirements and policies.

Uniform Credentials/Certification Infrastructure: Interdomain access requires, at a minimum, a common way of expressing the identity of a security principal such as an actual user or a resource. Hence, it is imperative to employ a standard (such as x.509v3) for encoding credential for security principals.

Support for Secure Group Communication: A computation can comprise a number of processes that will need to coordinate their activities as a group. The composition of a process group can and will change during the lifetime of a computation. Hence, support is needed for secure communication for dynamic groups. No current security solution supports this feature.

Support for Multiple Implementations: The security policy should not dictate a specific implementation technology. Rather, it should be possible to implement the security policy with a range of security technologies, based on both public and shared key cryptography.

Exportability: Grid Security requires the code to be developed such that it is exportable and executable in multinational testbeds.

There are additional security requirements which need to be considered when designing a Grid Security solution, such as firewall traversal, security assurance, accounting, auditing, user privacy, security management, policy management and so forth. Among these, mechanisms capable of allowing data to cleanly traverse firewalls without compromising local control of firewall policy is crucial for implementing cross-domain Grid systems.

This section summarized major security requirements in a Grid. The next section discusses the Grid Security issues.

1.2.2 Grid Security Issues

In the context of Grid Computing, Grid Security issues assume significant importance mainly due to the heterogeneity and high speed requirements of Grid applications. The differing security policies and sharing of heterogenous resources located in various multiple administrative domains also imposes some challenging issues [67], which can be broadly classified into:

1.2.2.1 Architectural Level Issues

Architecture level issues address the concern of the Grid system as a whole. Issues like information security, authorization and service level security generally destabilize the whole system and hence an architecture level solution is needed to prevent those. These issues are briefly discussed here:

Information Security: It can be defined as the security related to the information exchanged between different hosts or between hosts and users. The concerns at the information security level of the Grid can be broadly described as issues pertaining to secure communication, authentication and issues concerning single sign on and dele-

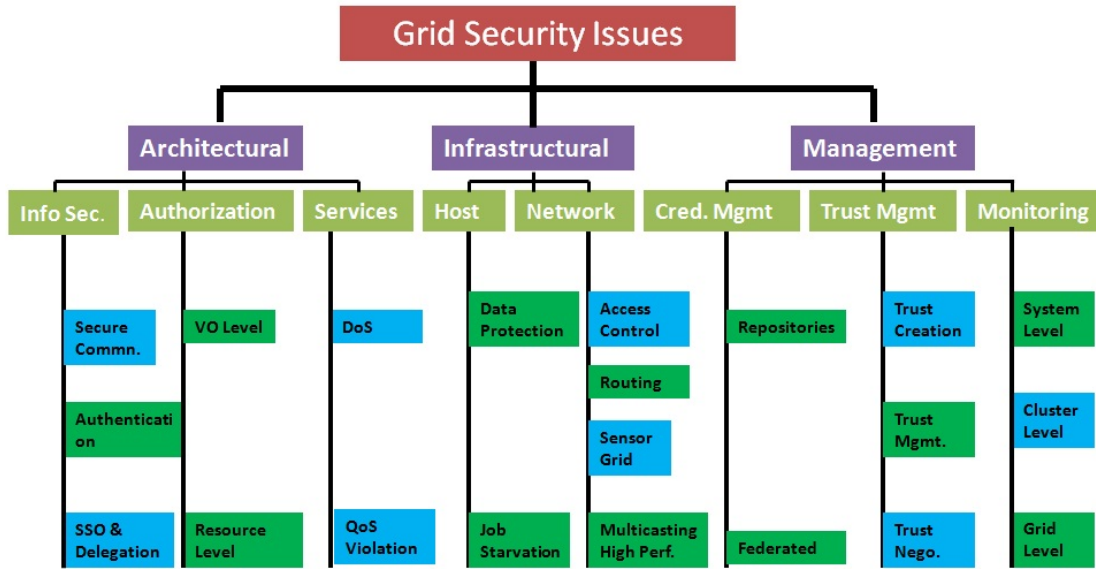


Figure 1.3: Grid Security Issues [67]

gation. Secure communication issues include those security concerns that arise during the communication between two entities. These include confidentiality and integrity issues. Confidentiality indicates that all data sent by users should be accessible to only “legitimate” receivers and integrity indicates that all data received should only be sent/modified by ‘legitimate’ senders. There are also issues related to authentication, where the identities of entities involved in the overall process can be accurately asserted. These are critical issues in all areas of computing and communication and become exceedingly critical in Grid Computing because of the heterogeneous and distributed nature of the entities involved. In addition to the secure communication features users are also concerned about single sign on capability provided by the Grid Computing infrastructure.

Authorization: Another important security issue is that of authorization. Like any resource sharing system, Grid systems also require resource specific and system specific authorization. It is particularly important for systems where the resources are shared between multiple departments or organizations and department wide resource usage

patterns are pre-defined. Each department can internally have user specific resource authorization also. The authorization systems can be mainly divided into two categories: Virtual Organization Level Systems and Resource Level Systems. VO level systems have a centralized authorization system which provides credentials for the users to access the resources. Resource level authorization systems, on the other hand, allow the users to access the resources based on the credentials presented by the users.

Service Security: One of the most important security threats existing in any infrastructure is the malicious service disruption created by adversaries. Many such examples exist in the Internet space where servers and networks are brought down by a huge amount of network traffic pumped in by adversaries and users were denied access to certain Internet based services. Since Grid computing deployment has not reached the “critical mass” yet, the service level attacks are currently nonexistent. The Grid service level security issues can be further subdivided into two main types: Quality of Service (QoS) Violation Issues and Denial-of-Service (DoS) related issues. The first issue is about the forced QoS violation by the adversary through congestion, delaying or dropping packets or through resource hacking. The second one is more dangerous where the access to a certain service is denied.

1.2.2.2 Infrastructure Related Issues

These issues are related to the host and network components which constitute the Grid infrastructure.

Host level security: Host level security issues are those issues that make a host apprehensive about affiliating itself into the Grid system. The main sub issues here are: data protection and job starvation. Whenever a host is affiliated to the Grid, one of the main concerns is regarding the protection of already existing data in the host. The

concern stems from the fact that the host submitting the job may be untrusted or unknown to the host running the job. To the host running the job, the job may well be a virus or a worm which can destroy the system. This is called the Data protection issue. Job starvation refers to a scenario where jobs originating locally are deprived of resources by Grid jobs scheduled on the host as part of the Grid system.

Network Security Issues: In the context of Grid Computing, network security issues assume significant importance mainly due to the heterogeneity and high speed requirements of many Grid applications. Moreover, the Grid inherits some of the generic network issues also. Access control and isolation are important requirements for traffic flowing through the Grid networks. In this area, integration of Grid technologies with Virtual Private Network (VPN) and firewall technologies assume significance. Routing of packets in networks based on routing tables is a specific network issue. Attacks in routing include link and router attacks which may cause significant destruction. Multicasting is an efficient means of information dissemination and may assume importance for Grid networks in the future. Member authentication, key management and source authentication are specific security issues in multicasting.

1.2.2.3 Management Related Issues

The Grid management is important as the Grid is heterogeneous in nature and may consist of multiple entities, components, users, domains, policies and stake holders. The different management issues that Grid administrators are worried about are credential management, trust management and monitoring related issues.

Credential Management: Management of credentials becomes very important in a Grid context as there are different systems which require varied credentials to access them. Credential management systems store and manage the credentials for a variety of sys-

tems and users can access them according to their needs. This mandates for specific requirements from the credential management systems. For typical Grid credential management systems mechanisms should be provided to obtain the initial credentials. This is called the initiation requirement. Similarly, secure and safe storage of credentials is equally important. In addition, the credential management systems should be able to access and renew the credentials based on the demand of the users. A few other requirements which are important for Grid systems are translation, delegation and control of the credentials. Based on the above requirements, credential management systems are mainly of two types: credential repositories or credential storage systems and credential federation systems or credential share systems. The first set of systems are responsible for storing credentials while the second set of systems are responsible for sharing credentials across multiple systems or domains.

Trust Management: Another important management issue which needs to be addressed, is the issue of managing trust. Managing trust is not unique to digital or computing systems as it is used everyday and in every sphere of life. Trust is a multi-dimensional factor which depends on a host of different components like reputation of an entity, policies and opinions about the entity. Managing trust is crucial in a dynamic Grid scenario where Grid nodes and users join and leave the system. Therefore, there must be mechanism to understand and manage the trust levels of systems and new nodes joining the Grid. Trust Management issue is undertaken as research area to be addressed in this thesis work.

Monitoring: Monitoring is the third and one of the most crucial management issues that needs to be tackled in a Grid scenario. Monitoring of resources is essential in Grid scenarios primarily for two reasons. Firstly, different organizations or departments can be charged based on their usage. Secondly, resource related information can be logged

for auditing or compliance purposes. The different stages of monitoring are: data collection, data processing, data transmission, data storage and data presentation. The data collection stage involves collecting data through different sensors located at different collection points. The gathered data can be static in nature like network topology, machine configuration or dynamic like CPU and memory utilization, system load, etc. The Data processing stage processes and filters the data based on different policies and criteria from the data collected from the sensors. The Data transmission stage involves the transmission of collected and processed data to the different entities interested. Transmission involves sending the data in a format understood by other parties over a transmission medium, for example the network. There may be a need for storage of gathered or processed data for future references which is carried out in the data storage stage. Finally, the data presentation stage presents the data in a format understood by the different interested entities. Grid security issues that need to be dealt with so as to secure Grid environments have been covered in this section while next section deals with challenges in Grid security that are of paramount importance and identified by OGF as essential and foremost, to be addressed.

1.2.3 Challenges in Grid Security

To make wide adoption of Grid a reality, there are various security challenges that need to be addressed. There are three most important Grid security challenges as identified by OGF which need to be tackled for ultimate success of Grid Computing as a powerful and secure distributed computing paradigm [64]. These three challenges are Integration, Interoperability and Trust Management. Out of these, two Grid Security challenges: Interoperability and Trust Management have been chosen to be addressed in this thesis work for making Grid as envisioned, a single and secure massive infras-

tructure - a Reality. These challenges have been discussed in detail in the next chapter.

However, a brief introduction has been presented here:

1.2.3.1 Integration

Integration challenge requires security architecture to be implementation agnostic. There are numerous security frameworks, standards and implementations available today. The majority of organizations and individuals have their own preferences about the security requirements that are most suitable for their own environment. This places burden on the participants to honor the existing security frameworks and/or seamlessly integrate with them. So, security architecture should be ‘implementation agnostic’ so that it can be instantiated in terms of existing security mechanisms. Security architecture should be ‘extensible’ also so that it can incorporate new security services when available and capable of integration with the existing security services.

1.2.3.2 Interoperability

Although interoperability intuitively can be understood from a high-level perspective, the concept has no single agreed upon definition. IEEE defines interoperability as “The ability of two or more systems or components to exchange information and to use the information that has been exchanged” [69]. This definition captures the intuitive understanding that the interoperable systems must be able to communicate in a meaningful way. Depending on the view on Grid, Grid interoperability can be interpreted differently. The different types of interoperability include internal interoperability within a Grid infrastructure, between Grid infrastructures and between Grid infrastructures and other systems.

At **protocol level**, different domains exchange messages across their protocol layers

so they need to have interoperability at each layer of the protocol stack. Mechanisms should be there to allow domains to exchange messages in a platform neutral way.

At **policy level**, secure interoperability requires that each party should be able to specify policy wishes, in order to engage in a secure conversation. The policies expressed by different parties should be mutually comprehensible.

At **identity level**, mechanisms are required to identify a user from one domain in another domain. For any cross domain invocation to succeed in a secure environment, the mapping of identities and credentials to the target domain identity is also required. Some approaches available for interoperability are discussed below.

Use Case Scenario: The user, who is registered in all the needed Grid-infrastructures, manually sends a task to proper infrastructure, controls the task's flow, receives, processes and combines the results.

Parallel Deployment: It is a modified version of use-case scenario. The difference lies in presence of a single entry point to several Grid infrastructures. This simplifies user's work but all the actions required are to be handled, manually.

Gateways: Gateway is usually a program module, which works on top of Grid-middleware, which can convert a task's description from one middleware format to another middleware format and send this converted task to another middleware, control its flow and receive its results. Gateway uses the interface of underlying middleware, so, from user's point of view, it seems possible to work with several middleware using the system of command of just one middleware. Gateway doesn't have features of advanced planning and combining of results. It just puts the received task to another middleware.

Adapters and Translators: Adapter is a program which can communicate with different Grid middleware. Adapters have their own user interface and task description

format and can control many aspects of task's execution in Grid Infrastructure. However, translator can be a part of adapter, or it can be also built in middleware, only. Adapters have features of advanced planning or combining several tasks in one logical workflow.

All these approaches were developed before the work for creating unified Grid Interoperability standard was started. Standardization is considered to be the best step for Interoperability, but it is still in development state [70].

1.2.3.3 Trust Management

Trust Management challenge deals with trust related issues among Grid participants. Grid service requests can span multiple security domains. Trust relationships among these domains play an important role in the outcome of such requests. A service needs to make its access requirements available to interested clients so that they understand how to securely request access to it. The trust among the participants in a dynamic virtual organization is a complex thing to achieve and this trust must be evaluated for each session or request. This requires federation of trust relationships among the participants. The dynamic nature of the Grid in some cases can make it impossible to establish trust relationships, among sites prior to application execution. As the participating domains may have different security infrastructures therefore, it is necessary to realize the required trust relationships through some form of federation among the security mechanisms. Trust management is the activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships. Two main approaches, currently available for managing trust in Grid Computing environments are Policy based trust management system and reputation

based trust management system.

Policy based trust management: Policy based trust management deals with using policy languages and engines for specifying and reasoning trust. The goal is to determine whether or not an unknown user can be trusted based on a set of credentials and a set of policies. The different entities or components constituting the system exchange and manage credentials to establish trust relationships which are further refined based on certain predefined policies. Policy based trust is involved in making access control decisions. It is intended for systems where behavior is guided by complex rules and policies. Policy-based systems require a policy database and information to make the trust related decisions. Policy-based systems, mostly, have low message overheads as the message exchange is minimal. The main component of latency in policy-based systems is due to searching the policy database rather than computing complicated trust functions. Policy based systems require some amount of replication to provide better reliability. Policy based systems are more secure than reputation based trust systems.

Reputation Based Trust Management: Reputation based trust management system focusses on trust computation models capable of estimating the degree of trust that can be invested in a certain party based on the history of its past behavior. It is intended for distributed systems where a system only has a limited view of the information in the whole environment. New trust relationships and trust values are inferred based on the available information. The available information is based on the recommendations and experience of other users. Another approach to manage trust is by using Trusted Authority (TA) where the basic concept is same as that of a CA (Certificate Authority) but is specifically meant for dealing with trust. But, in large scale distributed computing system like Grid, its credibility and usage decreases and uncertainty increases as the community of its trustees grows.

Thus, it has been construed that to secure and enable wide adoption of Grid environments, key challenges to be addressed are interoperability and trust management. It is apparent that for establishment for secure Grids, interoperability and trust management solutions must be designed, developed and implemented.

Therefore, designing, developing and implementing solutions to address security challenges specific to Grid has been the research motivation behind this work. The next section briefly highlights contributions of the thesis.

1.3 Limitation with existing approaches

Grid Computing landscape is transforming and continuously evolving, posing complex Security requirements, issues and challenges at research community, to provide solutions for them. Most of the Security solutions available do not provide comprehensive models to address different types of Security requirements and access control policies as discussed in previous section. Many of them are not based on open standards. Some of them are problem specific or use proprietary mechanisms. As also observed during exhaustive review carried out it is clear that, no solution for Grid services exist that provides an answer to interoperability and trust management, as a whole. Therefore, the thesis contribute as providing various solutions for interoperability and trust management. These solutions are discussed briefly in next section and explained in detail in the thesis.

1.4 Thesis Contributions

The thesis contributes to provide solutions for security challenges faced in Grid Computing. The contributions of the thesis are mentioned here:

- (i) The thesis presents detailed literature review of the work done in the area of Grid Security and addresses challenges such as trust management and interoperability.
- (ii) The thesis identifies and proposes various interoperability solutions to realise the original of grid as a single massive infrastructure. The interoperability solutions proposed are Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE). TUX-INTERO is a Web Services based interoperability solution which discovers and integrates resources based on various platforms and middleware. TUX-INTEROP is a grid portal for secure interoperation of grids. PIE is grid portal which integrates resources from various middleware and allows access to end users at a single interface.
- (iii) The thesis identifies and proposes trust management solution for establishing secure grids as Thapar University Extensible-Trust Management System (TUX-TMS). TUX-TMS establishes and evaluates trust among untrusted and unknown partners in a Grid. This system provides incentive to resource/service providers to contribute resources and motivates resource/service users to interact and transact depending on their reputation and trustworthiness information. Thereby, the decision making in untrusted open Grid environments is achieved.
- (iv) The thesis presents design, development and implementation of proposed Interoperability and Trust Management Solutions.
- (v) The thesis integrates and tests TUX-INTERO and TUX-TMS on TUX-CITMS: A Grid Testbed, which spans across Thapar University Campus using several test cases. It also demonstrates the testing results of TUX-INTEROP and PIE using

TU Campus Grid using set of test cases.

- (vi) The thesis demonstrates the applicability of the proposed models in interoperating and securing transaction among multiple domains in a Grid environment.

The next section discusses organization of this thesis.

1.5 Thesis Organization

The Thesis has been organized into six chapters as discussed here:

With a brief introduction of Grid Security in Chapter 1, the rest of the thesis is organized as follows. Chapter 2: Literature Review - It focusses on Interoperability and Trust Management challenges faced by Grid, which are chosen as research objectives to be addressed. The third Grid security challenge: Integration is covered briefly. Interoperability and Trust Management issues such as authentication and authorization have been elaborated here. Further, this chapter discusses various solutions for Interoperability and Trust Management in Grids. An analysis of interoperability and trust management issues like authentication, authorization in various GCE's such as GLOBUS, UNICORE etc. has been presented here. Problem formulation has been discussed at the end of the chapter with laying objectives for the research work undertaken. It fulfils Objective 1 and 2 of the thesis.

Chapter 3: The Proposed Interoperability Solutions: TUX-INTERO, TUX-INTEROP and PIE - It presents Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE) as interoperability solutions in Grids. TUX-INTERO is a Web Services based interoperability solution which discovers and integrates resources based on various platforms and middleware. TUX-

INTEROP is a grid portal for secure interoperation of grids. PIE is grid portal which integrates resources from various middleware and allows access to end users at a single interface. This chapter discusses taxonomy, architecture and implementation details of the proposed Interoperability Solutions. The Interoperability solutions proposed fulfil partially third objective of the thesis and broadly it fulfills the original vision of Grid Computing aimed at providing a single massive infrastructure to users instead of disparate resource islands.

Chapter 4: The Proposed Trust Management Solution: TUX-TMS - It presents Thapar University Extensible-Trust Management System (TUX-TMS) as trust management solution in grids. TUX-TMS is a flexible Reputation based Trust Management System for establishing trust and assessing trustworthiness of domains in Grid Computing environments. The taxonomy, parameters and evaluation parameters used for TUX-TMS have been covered in this chapter. The system architecture and components of TUX-TMS has also been discussed in detail. Implementation details are discussed at the end of the chapter. It partially fulfills Objective 3 of this thesis.

Chapter 5: Deployment, Testing and Validation of the proposed Solutions - It explains deployment details, experimental setup and testing results of the proposed Interoperability and Trust Management solutions. TUX-INTERO and TUX-TMS have been integrated, deployed and tested on TUX-CITMS. TUX-CITMS, a Grid Testbed has been specifically developed for testing purposes. This testbed spans across whole Thapar University Campus including entire academic area, some part of residential areas, utility area like library, cyber surfing labs, cafeteria etc. Testing results have been presented in the form of sixteen test cases. Every test case defines a testing criteria, test inputs, expected outputs and finally validates the outcome to be correct/ as expected. TUX-INTEROP and PIE have been deployed and tested further on Thapar Univer-

sity Campus Grid which comprised of Computer Science and Engineering Department labs. Testing Results of TUX-INTEROP and PIE have been demonstrated in form of four and five test cases, respectively. The testing results clearly demonstrate that the proposed solutions are working and can be effectively used to address interoperability and trust management challenges to establish secure Grids. It has been established from the experiments that this approach gives approximately secure transactions to new and unknown users. This also results in enhanced Grid usage while addressing interoperability and trust management challenges in a Grid Computing Environment. It fulfills Objective 4 of the thesis.

Chapter 6: Conclusion and Future Directions - This chapter concludes the work done with a brief discussion on challenges faced for designing systems for establishing secure Grids. In a Grid, Interoperability systems need to provide solutions at multiple levels which is achievable through standardization. It highlights the main contributions of this research work and specifically focusses on the future work that can be undertaken to extend the work presented in this thesis.

To conclude, this chapter discussed evolution of Grid Computing and its usage today as a powerful distributed computing paradigm. It provides an overview of Grid Computing fundamentals while traversing through components, architecture and standards of Grid Computing. The total emphasis has been laid on Grid Security Component which has been chosen as research motivation. Last but not the least, thesis organisation is discussed here.

Chapter 2

Literature Review

This chapter analyzes existing work in the area of Grid Security challenges, present gaps in current research, formulates problem statement and introduces the objectives of the thesis. This chapter is divided into various sections. Section 2.1 discusses existing Grid security technologies and standards. Section 2.2 explores key security challenges in Grid Computing Environments such as Integration, Interoperability and Trust Management, which are identified as research motivation and work. Section 2.3 presents current state of Interoperability and Trust Management issues such as authentication and authorization in various Grid Computing Environments. Section 2.4 discusses current state of Security in popular Grid middleware.

2.1 Grid Security Technologies and Standards

Grid Security research currently focuses on creating building blocks that can provide a secure distributed infrastructure for the highly sensitive applications and commercial implementations of existing and next-generation Grids. To do this, researchers are defining protocols, technologies and standards that make it possible to build Secu-

rity overlays across organizational infrastructures to enable a trusted environment for resource sharing. These Security technologies and standards are discussed here [63].

2.1.1 Grid Security Infrastructure (GSI)

Most current Grids use Grid Security Infrastructure (GSI), developed as a part of the Globus toolkit. The GSI uses public key cryptography (also known as asymmetric cryptography) as the basis for its functionality. The primary motivations behind the GSI are the need for secure communication (authenticated and perhaps confidential) between elements of a computational Grid, the need to support security across organizational boundaries, thus prohibiting a centrally-managed security system and the need to support “single sign-on” for users of the Grid, including delegation of credentials for computations that involve multiple resources and/or sites [71]. GSI has been discussed in detail in the Chapter 2.

2.1.2 Message-level security

Transport-layer security [72] relies on securing the transport mechanism itself, on which message is being passed. Transport Layer Security (TLS) channels typically make use of both symmetric-key and public-key cryptography. When establishing a channel between entities authentication is performed using public keys. This may be one-way authentication in the case where only the client authenticates the server or it may be two-way mutual authentication in which both parties authenticate each other. A peer signs some message to prove possession of the private key associated with a particular public key tied to its identity. After successful authentication, a symmetric session key is shared between the peers to take advantage of the lower computational overhead of symmetric-key ciphers. This is the most common form of authenticated communication

in current Grid middleware. With the merging of Grid and Web services, Grids are transitioning to use message-level security. Because message-level security works by individually securing each Simple Object Access Protocol (SOAP) message, it can work with any transport-layer protocol and it allows the implementation of different levels of protection depending on the sensitivity of the data.

2.1.3 WS-Security

IBM, Microsoft and VeriSign have submitted to Organization for the Advancement of Structured Information Standards (OASIS), a Web Services (WS) Security specification called WS-Security [73]. Figure 2.1 presents framework of WS-Security specifications. The framework consists of different security related specifications. WS-Security was the first specification, which was released. Later, the specifications WS-Trust, WS-Policy, WS-SecureConversation and WS-Federation were released. WS-Security describes how signature and encryption headers are attached to SOAP messages. In addition, it also describes how to attach security tokens, including binary security tokens such as X.509 and Kerberos tickets, to messages. WS-Policy provides a general purpose model and corresponding syntax to describe the policies of a Web Service, for example, required security tokens, supported encryption algorithms and privacy rules etc. WS-Trust describes a framework for trust models that enables Web Services to securely interoperate. It provides a framework for requesting and issuing security tokens. WS-Privacy is not yet available but will describe a model for how Web Services and requesters can state subject privacy preferences and organizational privacy practice statements. WS-SecureConversation describes how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys. WS-Federation describes how to manage and broker the trust relationships

in a heterogeneous federated environment including support for federated identities. WS-Authorization is not yet available but will describe how to manage authorization data and authorization policies. Although, WS-Security does not in itself provide a

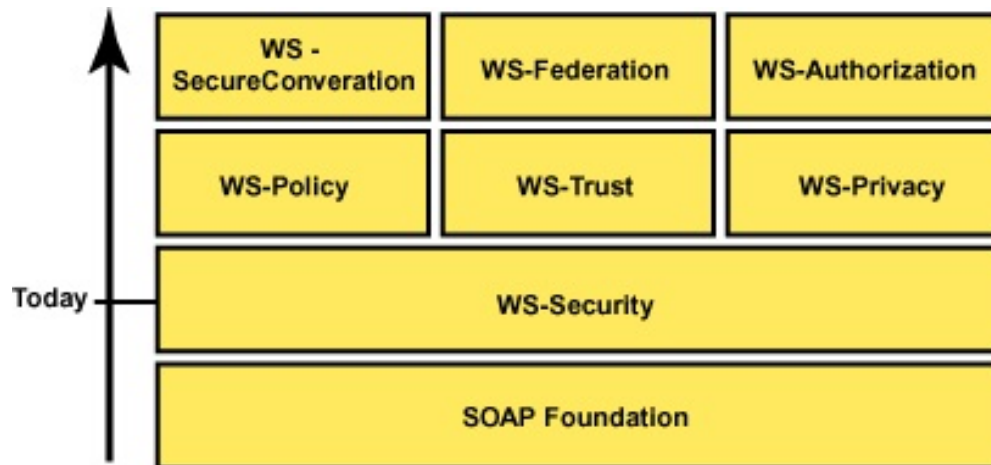


Figure 2.1: Web Services Security Specifications [4]

complete security solution, it defines building blocks that Web services developers can use to build security frameworks in the higher layers. Thus, Grid security architects are using WS-Security as a basis for constructing a Grid Security fabric that spans diverse security models [28].

2.1.4 Security Assertion Markup Language (SAML)

When organizations share resources in a Grid, they need a common language with which the Grid entities can exchange security information [74]. Security Assertion Markup Language (SAML) is a standard proposed by OASIS, which defines a language and protocol to exchange authentication and authorization information. SAML assertions contain information about authentication references, authorization decisions and attributes associated with a specified entity. SAML policy can reside in external

policy stores, making it easy for the virtual organization to use various existing policies in the local domains. SAML defines a query-reply protocol interface that lets clients request assertions from SAML authorities. This protocol, consisting of XML-based message formats, can bind easily to many different underlying communications and transport protocols. SAML currently defines one binding to SOAP over HyperText Transfer Protocol (HTTP). In addition, time stamps associated with SAML assertions and queries let Grid administrators associate temporal constraints with the state of virtual organization and user attributes, thus mirroring the dynamic trust in Grid environments.

2.1.5 Extensible Access Control Markup Language (XACML)

XACML is an initiative to develop a standard for access control and authorization systems [75]. Most of the current systems implement access control and authorization in a proprietary manner. XACML provides a policy language which allows administrators to define the access control requirements for their resources in a standard and portable way. It also provides a mechanism that offers much finer granular access control than simply denying or granting access. XACML architecture is tightly intertwined with SAML architecture. While SAML addresses authentication and provides a mechanism for transferring authentication and authorization decisions between cooperating entities, XACML focuses on the mechanism for arriving at those authorization decisions. The importance of XACML is that it can be used to express policies for different components of the same organization. Being able to express policies of different components in the same way is a big advantage for any organization.

Another Security model for Grid Services is proposed by Open Grid Services Architecture (OGSA) Security Workgroup [64]. In addition to describing unique Grid

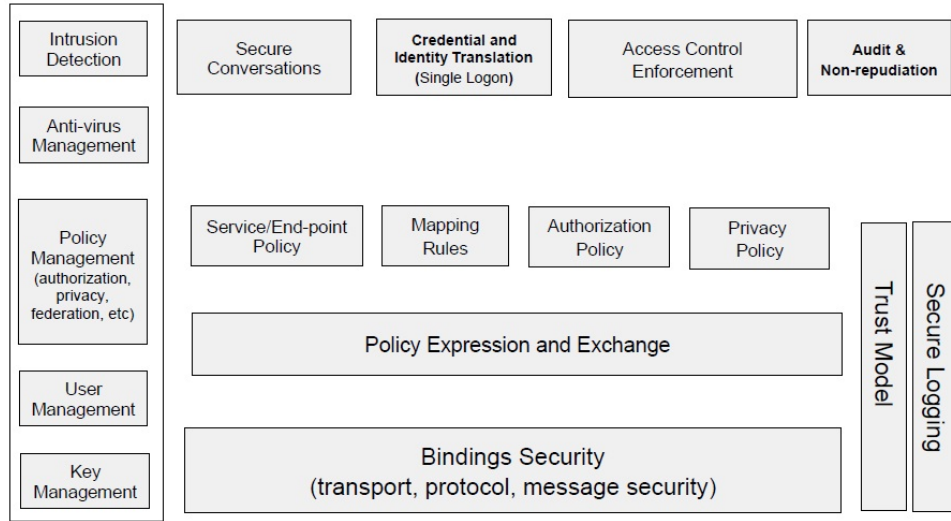


Figure 2.2: Components of Grid Security Model [64]

Security requirements, it presents a layered architecture for addressing different Security related issues within OGSA (as shown in Figure 2.2). It defines a comprehensive Grid Security architecture that supports, integrates and unifies popular Security models, mechanisms, protocols, platforms and technologies in a way that enables a variety of systems to interoperate securely. This Security architecture is intended to be consistent with the Security model that is currently being defined for the Web Services Framework [64]. The base layer of this model is binding Security that deals with transport mechanisms. In this model, at application level there are specific components such as secure conversation, credential identity translation, access control enforcement and audit and non-repudiation depends on policies and rules for authorization, privacy, identity/credential mapping and service/end point. In order to apply and manage these policies, one needs a language for policy expression and exchange and means to securely communicate through bindings to transport protocols. To realize the functionality of this layer, WS-Policy or XACML specifications can be used to express and exchange policies among different entities. One side of this model has trust model that defines

trust anchors and how trust is derived. Other side of this model has management components that are also subject to policy enforcement. This framework proposes a layered architecture but does not provide specific models to handle authentication related issues, interoperability, trust management, authorization and access control related issues. The models are proposed to lay foundation for next generation Grids.

2.2 Grid Security Challenges

The real power of Grid can be harnessed only if it provides secure access to resources/services. Grid setup requires a secure environment where users have access to resources precisely on the basis of their rights and where the system has record of their activities which they later cannot deny. The traditional approaches to address Security requirements in distributed systems do not work for Grid systems as Grids present entirely new Security challenges discussed in this section. The key Security challenges as also identified by OGF, faced in a Grid Computing environment can be grouped into three categories: integration with existing systems and technologies, interoperability with different hosting environments (e.g., J2EE servers, .NET servers, Linux systems) and trust relationships among interacting hosting environments [64]. As integration doesn't fall in the premise of this thesis, therefore brief literature review of significant work is presented followed by extensive review of work on Interoperability and Trust Management challenge.

2.2.1 Integration

In the past, there have been various initiatives from a number of research groups towards addressing Integration in Grid Computing. In Grid scenarios, where it is favorable to operate multiple different implementations, for example, different middleware, in parallel or where there is no agreement on or knowledge of a common standard, non-intrusive components, bridges, become necessary to enable an integration. Hegering et. al. [76] describes three kinds of architectural bridges which can be applied for enabling Integration in Grids. These three types of bridges are multi-architectural platforms, management gateways and multi-architectural agents. In this work, integration can be realized by bridging communication (e.g. interfaces), information (e.g. data description schema), organization (e.g. roles) and functional models (e.g. queries).

The first Integration project [77] was carried out jointly between the University of Kent and the Globus Team, which integrated Globus Toolkit and PERMIS [78], so that Globus Toolkit 4 (GT4) could replace grid-mapfile authorization with PERMIS policy-based authorization. This was enabled by specifying an open protocol, under the auspices of the Global Grid Forum (GGF), which allows a Grid application to make a call to any external Policy Decision Points (PDP). In this project, PERMIS [78] was also modified so that it can now run as a standalone external decision engine as well as an integrated internal one.

The second project, SIPS [79], again at the University of Kent, integrated PERMIS with Shibboleth [80], so that a Shibboleth resource can utilize the PERMIS policy-based authorization infrastructure. The third project, GridShib [81], [82], run by the Globus team, allowed Globus Toolkit and Shibboleth also to interoperate. In this project, a pair of software plugins, one for Globus Toolkit and another for Shibboleth, enabled a Globus Toolkit Grid Service Provider (SP) to securely request user attributes

from a Shibboleth Identity Provider (IdP).

In parallel with the GridShib project, the team at Kent integrated PERMIS with GridShib, so that Globus Toolkit 4 can now call PERMIS through Security assertions [83]. The final piece in the integration jigsaw puzzle was to allow a user who does not have a public key certificate to run a Grid job using his normal organizational authentication credentials. The University of Manchester is performing this integration in their SHEBANGS project [84]. A key component of their design is a new component called the Credential Translation Service (CTS), which behaves like a standard Shibboleth SP in the Shibboleth world and an online Certificate Authority and MyProxy client in the Grid world.

Marcos et.al. [85] proposes integration of Enterprise and Global Grids by proposing the integration of Gridbus Broker with diverse enterprise grid middleware technologies including XGrid, Portable Batch System (PBS), Condor and Sun Grid Engine (SGE). These Integration projects aimed at federating credentials for providing users from one platform an access to resources in other platform in a convenient manner. TrelisWeb [86] provides an interface that address resource scheduling and allocation, single log on and access control [87]. The proposed system is integrated with PBS, SGE and IBM's LoadLeveler [88] and leverages these tools to a metacomputing scenario. However, this work does not contemplate integration with global Grid middleware such as Globus and Unicore.

Grid(Lab) Grid Application Toolkit (GAT) [89] presents an architecture in which common APIs sit between Grid applications and diverse Grid middleware. The main goal of GAT is providing Grid programmers with a uniform interface to different middleware and services, such as data catalog, data replication, data movement, Globus Grid Services and Globus 2.X and 3.X Pre WS [71]. By providing such API, GAT aims at

enabling the easy development of “Grid-aware” applications. This API is targeted to the development of portals and so can enable a range of Grid applications. The next section explores Interoperability challenge in a Grid Computing Environment.

2.2.2 Interoperability

The interoperability problems within contemporary Grid environments are the motivation behind substantial efforts from both research and infrastructure projects. These efforts can be divided into two classes, short-term solutions to provide interoperation between existing infrastructures and more sustainable approaches based on adoption of standard data formats and interfaces. Some of the work done are reviewed here :

The first step taken towards Interoperability was by providing interoperability between the main ARC-based Grid-infrastructure and Globus [71] where GridWay [55] was chosen as an adapter that can help to provide such interoperability, which interfaces resources running one of these infrastructures: GT2, GT4, and LCG. In 2001, an architecture and prototype implementation for a Grid-enabled computational framework based on Cactus was presented [90]. DISCOVER [91] by Vijay Mann and Manish Parshar provided collaborative access to high-performance parallel and distributed applications for interaction and steering using Web-based portals. DISCOVER collaboration is also considered as a first step towards achieving interoperability in Grid Computing.

Brooke [92] developed a more abstract approach to the problem of interoperability of Grid resource descriptions, by describing abstract spaces between which quantitative descriptions of requests for computational resource must be mapped to resource provision.

Another most generic way for achieving interoperability are Workflows which are widely

utilized by e-scientists for job/service orchestration and execution [93]. Cao et. al. developed GridFlow [94], a Grid workflow management system (WMS) at Warwick which worked on performance prediction and Grid resource management. It was also considered first step towards Workflows Management System, followed by P-GRADE portal [43], Pegasus [95] etc. The P-GRADE Portal gives a Globus-based implementation for the collaborative-Grid and collaborative-user concept [96]. One of the design goals of P-GRADE was to make it possible for a user to develop a workflow for the Grid without knowing the particulars of the underlying Grid technology and to be able to move the work to other Grids without altering the workflow. The focus is on allowing a user to work at a workflow level and interoperate with multiple middleware. Moreover, due to the multi-Grid concept a single portal installation can serve user communities of multiple Grids. These users can define workflows using the high-level graphical notations of the Workflow Editor, can manage certificates, workflows and jobs through the Web-based interface of the Portal Server. They can harness resources from multiple VOs and can migrate applications among Grids without learning new technologies or interfaces. It is predicted that, the integration of the most commonly used job submission, service invocation and data access mechanisms into Grid workflow solutions and the ability of these workflow systems to share computation and data between each other would solve most of the interoperation challenges e-Scientists faced today. Kacsuk [97] worked on workflow-oriented portal which enables the interoperability of various Grids during the execution of workflow applications and supports users to develop and run their Grid workflows in a collaborative way. Tamas and Timo [98] analyzed the aspects of integrating heterogeneous data resources to computational Grid workflows. They also presented reference implementations for Open Grid Services Architecture Data Access and Integration (OGSA-DAI) [99] and SRB [100] data resource integrations.

As a result of their work, the target workflow system was able to access all commonly used Grid file catalogues and also various relational and XML databases.

Automatic Deployment Service (ADS) [101] architecture enabled automatic deployment and migration of GEMMLCA Grid services to different sites within the same Grid domain. The combination of this architecture with the Interoperable Bridge Service (IBS) extended deployment and migration capabilities to different Grid domains.

The WorldGrid [102] transatlantic testbed provides Grid interoperability across Europe and United States of America domains by running a given set of core services and several collective optional services, sharing the same authentication and authorization mechanism and using a common schema for resource location and status in the context of a Virtual Organization.

Venugopal et al. [103] designed a broker that interoperates with various batch systems as well as multiple Grid middleware. In an approach suggested by Kertesz [104], interoperability is achieved on a resource broker level instead of a Grid middleware level. Rather than interfacing resources directly, their software interacts with the respective resource brokers of the used middleware. Kertesz also defines a language for communicating broker capabilities [105]. A similar approach is suggested by Rodero et.al. [105], that argue in favor of a metabroker based on their experiences from the HPCEuropa project. The proposed metabroker would have unified mechanisms for all tasks in the job management process and interact with Grid-specific brokers. Rodero et.al. also surveyed that to what extent existing and proposed standards fulfill the requirements of the metabroker. Bobroff [106] discussed different architectures for interoperable metascheduling and present a hybrid approach that combines hierarchical and peer-to-peer architectures. The contribution by Peirantoni et.al. [107] utilizes WSRF-based MetaGrid services as a bridge between users and different Grids. Set theory is used

to formally describe the MetaGrid services and their interactions. Notable examples of projects that focus on standardization include the Genesis II Grid system [108]. Genesis II supports a wide range of OGSA standards [109], including JSDL, OGSA-BES, OGSA Resource Namespace Service (RNS) [110] and OGSA-ByteIO [111]. A strong focus on usability allows users to interact with Genesis II resources through a set of common shell commands. An OGF report [112] shares implementation and interoperability experiences of JSDL and surveys projects that adopt this standard. An OGF report [113] by Newhouse discusses five scenarios related to the execution of applications on a cluster from an interoperability perspective. This report further illustrates how these scenarios can be realized using OGSA standards such as JSDL (with proposed extensions), OGSA-BES, the GLUE 2.0 [114] information model, RNS and Distributed Resource Management Application API (DRMAA). The purpose of an ongoing collaboration between the European Telecommunications Standards Institute (ETSI) [115] and the OGF is to ensure that the standards for communication within the Grid are based on requirements from both industry and research. The Open Middleware Infrastructure Institute (OMII) Europe [116] targets to make components for job management (OGSA-BES), data integration and accounting available for multiple platforms, for example, gLite [117], Globus [71] and UNICORE [25]. The UniGrids project [118] focuses on developing an OGSA-compliant infrastructure on top of the UNICORE software. Grimme et al. [119] analyze the teikoku scheduling framework in the light of standards-compliance and investigates the gaps between this framework and the proposed OGF Grid scheduling architecture. Daniel [120] proposed a simple model for transparent Grid interoperability which allowed the end user of one Grid, which is comfortable with that Grid environment, to submit jobs request within his Grid and have them propagated through other Grids.

For each pair of Grids, the worldwide LHC Computing Grid (WLCG) [121] identifies common elements in the software stacks and develops extra glue components to connect compatible elements. Currently, interoperability between EGEE and OSG is the most developed having bi-directional job submission is possible. OSG sites appear in the EGEE information system and are tested using EGEE test suites. It should be noted that many of the issues involved in providing ongoing interoperation between Grids are administrative rather than technical. While pragmatic, WLCG's approach is not scalable, since for N Grids it leads to $\approx N^2$ extra GLUE components, $\approx N^2$ interoperability solutions. Elmroth [122] investigated how principles of SOAs combined with standardization can be used to achieve adaptability in resource selection as well as Grid middleware independence have been investigated. As a proof-of-concept, Elmroth designed and implemented a general, service-based Grid resource brokering architecture and demonstrated the feasibility of the approach by integrating job submission service with the NorduGrid/ARC middleware [123].

Timo et al. [124] proposed a distributed architecture for an interoperable information service, which combines data unification and categorization with policies for VO membership, VO resource management and data transformations. This service builds the basis for an integrated and interoperating monitoring of Grids, which provide their data to more than one VO and utilize heterogeneous information services. A standardized interface for the exchange of state data between Grid components was proposed by the OGF, the Web Services Resource Framework (WSRF), which is already widely used. A semantic-based scalable decentralized Grid resource discovery framework has been proposed which, integrates ontology, Peer-to-Peer network and intelligent agents to build the framework. The framework consists of an ontology model, an agent model and a set of algorithms for implementing the P2P architecture and searching the shared

resources. The framework satisfies Grid resource discovery features such as scalability, decentralization, dynamism and interoperability [125].

The Open Grid Services Architecture (OGSA) High Performance Computing (HPC) Profile [109] is a proposed standard for Grid interoperability in High Performance Computing (HPC) environments. It references existing specifications including:

- OGF Job Submission Description Language (JSDL)
- OGF OGSA Basic Execution Service (BES)
- WS-I Basic Profile

The OGSA HPC Profile leverages standard Web services protocols and development environments. It allows application and middleware software providers to target multiple HPC systems via a single protocol. The Open Grid Forum's SuperComputing (SC06) [126] interop prototype demonstration showed interoperability between multiple 3rd party resource managers and Web services platforms using the OGF Open Grid Services Architecture HPC Profile. Demo stations were setup at booths throughout the conference where tasks were submitted to compute clusters via the OGSA HPC Profile. The interoperability between various Grid infrastructures was successfully demonstrated at SC06. The diagram 2.3 illustrates how the OGSA HPC Profile is leveraged to support various usage scenarios. The compute resources host a particular application that performs a processing function. In order to run the application, a user requests that a job be run through the resource manager that governs access. Each application is described using JSDL, an XML-based language for describing the job to be run, while the OGSA-BES Web services interface allows Web Services clients to access functions of a resource manager. There are many groups working on providing standards for Grid Interoperability such as The Grid Scheduling Architecture - Research

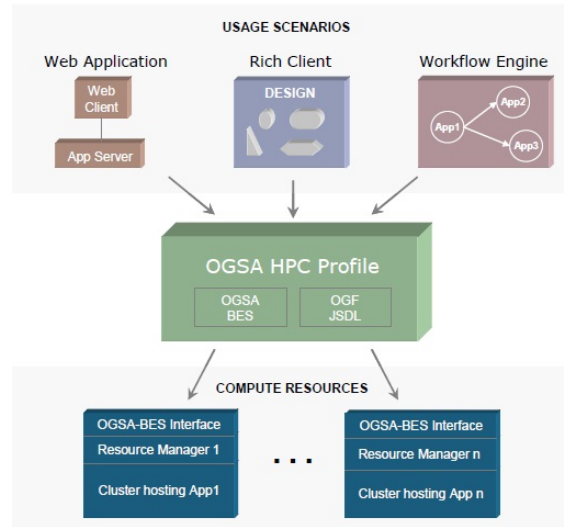


Figure 2.3: HPC Grid Interoperability Demonstration [126]

Group (GSA-RG) of OGF [27] is currently working on a project for enabling Grid scheduler interaction. They are trying to define common protocol and interface among schedulers to enable inter-Grid usage. In this work, they have proposed a Scheduling Description Language (SDL) to extend the currently available job description language solutions. This work is still in progress, up to now only a partial SDL schema has been created. The meta-scheduling project in Latin American (LA) Grid [105] aims to support Grid applications with resources located and managed in different domains. They define broker instances with a set of functional modules: connection management, resource management, job management and notification management. These modules implement the Application Programming Interfaces and protocols used in LA Grid through Web Services. Each broker instance collects resource information from its neighbors and saves the information in its resource repository or in memory. The resource information is distributed in the Grid and each instance will have a view of all the resources. Iosup et al. proposed a novel delegated matchmaking architecture (DMM) for interoperating Grids [127]. Their hybrid hierarchical/distributed architec-

ture allowed the interconnection of several Grids, without requiring the operation of a central point of the hierarchy. In DMM, when a user's request cannot be satisfied, remote resources are transparently added to the user's site, through delegated match-making.

The problem of Grid interoperability is much wider than the integration of existing middleware and in the new OGSA [28] framework there are other systems, such as Microsoft .NET and the Perl-based Open Grid Services Infrastructure- OGSI::Lite, that must be considered. There are other groups working towards designing solutions for Interoperability in Grids. One such group is The High Performance Computing (HPC) Profile Working Group (WG), an OGF group, co-chaired by Marty Humphrey of the University of Virginia and Steven Newhouse from EGEE which is part of the compute area within the OGF Standards function. The HPC Profile WG [109] is an effort in OGF to create the profile and protocol specifications needed to realize the vertical use case of batch job scheduling of scientific/technical applications. The group aims to establish interoperability solutions that would cover the key services such as Job Submission, Information Services, Storage Management, Accounting, Job Monitoring, Database Access and Virtual Organization Management. The cornerstone of interoperability for Job Submission is adherence to the Web Services-Grid Resource Allocation and Management (WS-GRAM), Job Submission Description Language (JSDL) and Basic Execution Service (BES) [128] standards. The initial target for Federated Information Services was to achieve full interoperability between popular Grid environments such as EGEE [32], OSG [31] and NAREGI [129]. The effort for Storage Management standardization focuses on the interoperability of the Storage Resource Broker (SRB) and Storage Resource Manager (SRM) products [130]. A generally accepted standard for Data Movement is GridFTP and for authorization is the Virtual Organization Mem-

bership Service (VOMS) [131] specification. A considerable effort was put in by several researchers in [132], [133], [134], [135], [136] [137] for providing solutions to interoperability and surveys were also reported in [138], [139]. But, still these efforts aimed at short term solutions. Clive proposed standards for interoperable Grids [140].

There have been specific efforts aimed at bilateral sharing and interoperability such as those between Ninf [141] and NetSolve [142], these have been made possible because of joint development efforts by their respective development teams. Laurence [143] presents a number of techniques that can be applied in the most important areas of interoperation and advocate standardization as the best way to achieve interoperability. Unfortunately, despite numerous efforts by Grid community to address interoperability, but, still current Grid resources exist as non-interoperable “production Grid islands”. It causes difficulties when the experiment overgrows the capabilities of one specific production Grid. These different production Grids are based on different Grid middleware, use different tools and policies for authentication and authorization, describe and submit jobs differently and provide data services based on a variety of protocols and access mechanisms. The utilization of resources spreading over multiple production Grids is not supported due to the heterogeneous and non-interoperable nature of current Grid middleware solutions. Standardization has been proposed as a long term solution for the interoperability problem. The Grid Interoperation Now (GIN) Community Group [144] of the Open Grid Forum (OGF) defines Grid interoperability as the native ability of Grids and Grid middleware to interact directly via common open standards [145]. OGF’s Grid Interoperability Now (GIN) [144] has identified four interoperability services that they would like to realise in the short term: authentication/authorization, a resource schema, job submission and data management. In the longer term, workflow, co-scheduling and accounting are also envisaged. The GIN

group aims to tackle interoperation, which they define as ‘what needs to be done to get Grids working together’, using existing software only. In contrast they consider interoperability to be the more general ability of software and hardware from various vendors to communicate.

Unfortunately, standardization and the widespread utilization of these standards is a long and painstaking process where success is not always guaranteed. Grid end-users want to utilize multiple production Grids and heterogeneous resources even today. They do not want to wait for the standardized Grids where all this happens by default. What they need in the short term are quickly implemented solutions that make current Grid tools and technologies, all work together. The GIN defines these short term efforts using existing technologies as Grid interoperation.

However, there still exists lot of work to be undertaken for achieving true interoperability in Grids. Although interoperability has been identified as a central issue for Grid based systems in previous works still there has been limited progress towards achieving this goal. These efforts only further highlight the benefits and significance of interoperability and the need for having a general solution for interoperability.

So far all attempts, to provide interoperability have been quick and short term solutions. To achieve interoperability, using the method of defining and adopting common open standards and architectures is a common approach, which relies heavily on standardization and implementation processes. Therefore, Interoperability has been chosen as challenge to be addressed in this research work by using standardized tools available. The next section unveils Trust Management challenge in a Grid Environment.

2.2.3 Trust Management

Grid systems involve the risk of executing transactions without prior experience and knowledge about each other's reputation. Managing trust is crucial in a dynamic Grid scenario where Grid nodes and users join and leave the system. There must be mechanisms to understand and manage the trust levels of systems and new nodes joining the Grid. Recognizing the importance of trust in such environments, there is a necessity of designing strategies and mechanisms to establish trust in Grid environment. Trust management is one such activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships [67].

Trust Management Systems: A trust management system generally includes negotiation of trust when a member joins, storage of trust metrics and distribution of trust metrics. There are two main approaches, currently available for managing trust in Grid Computing environments as elaborated here:

Policy based trust management Most policy based trust management systems focus on protocols for establishing trust in a particular context. Some make use of a trust policy language to allow the trustor to specify the criteria for a trustee to be considered trustworthy. In this section, a few of these solutions, namely: Public Key Certificates, PICS, PolicyMaker, KeyNote, REFEREE, SD3, Fidelis, TrustBuilder, TCPA, Poblano and a few other emerging trust management solutions are discussed.

In a Public Key Certificate [146], a digital certificate is issued by a certification authority and verified that a public key is owned by a particular entity. The certification authority does not vouch for the trustworthiness of the key owner, but simply authenticates the owner's identity. However, the policy governing what resources or services the trustee is permitted to access is not handled by the certificate infrastructure, but

is left up to the application. PGP and X.509 are main certificate systems dealing with authentication.

The PGP trust model [146] used for authentication relating to electronic mail type of applications between human users supports a ‘web of trust’ model. In PGP, there is no centralised or hierarchical relationship between certification authorities as with X.509. The X.509 trust model [147] [148], a strictly hierarchical trust model for authentication assumes that certification authorities are organized into a universal ‘certification authority tree’ and that all certificates within a local community will be signed by a certification authority that can be linked into this tree [149]. Due to PGP’s lack of official mechanisms for the creation, acquisition and distribution of certificates it is considered unreliable for Grids, but appropriate for personal communication. X.509’s rigid hierarchical structure may lead to unnatural business alliances between competing companies that violate the natural order of trust. Some applications, such as the reference information distribution systems need certificates to have a lifespan longer than is currently allowed by either scheme. Additionally, current PKI implementations contain no systematic and reliable method of obtaining evidence about entities involved in an online transaction [150].

PICS [151] was developed by the World Wide Web Consortium (W3C) as a solution to the problem of protecting children from pornography on the Internet. The basic idea behind PICS is that there needs to be a filter between the potential viewer and web documents. PICS defines standards for the format and distribution of labels, which are meta-documents describing web documents. The effectiveness of the PICS framework lies in the expressiveness of the filtering languages and the quality of the rating services.

PolicyMaker [149], a trust management application, developed at AT&T Research

Laboratories specifies what a public key is authorized to do. It evaluates whether a proposed action is consistent with local policy. PolicyMaker uses the credentials given to it to prove that the requested action complies with the policy (this process is referred to as compliance checking). An application gives the PolicyMaker engine, a (set of) requested action(s), a set of credentials and a policy and the engine tries to prove that the credentials contain a proof that the requested action(s) complies with the policy. KeyNote [152] is the successor to PolicyMaker and was developed by AT&T Research Laboratories. It has the same design principles of assertions and queries but includes two additional design goals, namely: standardization and ease of integration. KeyNote's predefined assertion language allows simpler integration with its compliance checker. Neither system addresses the problem of how to discover that credentials are missing and neither system supports negative assertions. The authors claimed that both these systems are a more general solution to the trust management problem than public-key certificates. However, they address only authorization based on public keys, which still does not comprehensively cover the entire trust management problem.

REFEREE [153], a trust management system for making access decisions relating to web documents developed by Yang-Hua Chu, is based on PolicyMaker. It considers a PICS label as the stereotypical web credential and uses the same theoretical framework as PolicyMaker to interpret trust policies and administer trust protocols, which are represented as software modules. Like PolicyMaker and KeyNote, REFEREE is a recommendation-based, query engine so it needs to be integrated into a host application. It evaluates requests and returns a tri-value and a statement-list, which is the justification for the answer. A tri-value is either true, false or unknown. True means 'yes, the action may be taken because sufficient credentials exist for the action to be approved', false means 'no, the action must not be taken because sufficient credentials

exist to deny the action' and unknown means 'the trust management system was unable to find sufficient credentials either to approve or deny the requested action'.

SD3 [154] is a trust management system that uses logic to represent security policies. It consists of a high-level policy language, a local policy evaluator and a certificate retrieval system. SD3 is an extension of work done by Gunter and Jim on QCM, which is a system for automatic certificate management. Like PolicyMaker, KeyNote and REFEREE, SD3 is a recommendation-based query system. Unlike these other systems, SD3 uses a logic-based language as its default notation. SD3 suffers from the same problems as KeyNote does. However, though the treatment of negative credentials is not mentioned in current literature on SD3, it may be theoretically able to handle some negative credentials due to its use of datalog.

Fidelis [155] is a policy-driven trust management framework which originates from the work on Open Architecture for Secure, Internetworking Services (OASIS), which is a role-based architecture for distributed authorization management. In [155], Yao states that the Fidelis policy framework is an abstract, conceptual foundation, which is used as a starting reference point for the implementation of a policy language.

Trustbuilder [156] [157] presents a policy language and infrastructure for trust negotiation for open systems. In TrustBuilder protocol and architecture, the negotiating parties establish trust between themselves by negotiating trust in a need-to-know manner. In this way, all the credentials are not disclosed to the either party. As it is a policy based trust management system it does not make use of any trust function to calculate trustworthiness of different parties. The TrustBuilder framework is focused on trust establishment, but does not include facilities for handling credentials regarding negative assertions and assumes nothing about the consistency of the credential expressions.

TCPA [158], a consortium of companies such as Compaq, HP, IBM, Intel and Microsoft was formed to make the computer platform trustworthy. The motivation behind the decision to create an alliance was the realization by these companies that customers viewed their products as not worthy of their trust. The TCPA Subsystem consists of two basic building blocks, namely: a Trusted Platform Module (TPM), which is a hardware instantiation of the TCPA specification and software to perform integrity metrics, in conjunction with the TPM. The framework may inadvertently infringe on end user privacy and give consortium members an unfair business advantage (the ability to effectively shut out any rival software by labeling them from an untrustworthy source). It is not clear from the literature whether the actions of trusted applications are constrained or even if proof of trustworthiness is done with respect to a particular (set of) task(s) that the application wants to perform.

Poblano [159], was an attempt by Chen and Yeager of Sun Microsystems to build a decentralized trust model based on the JXTA platform. Poblano is based on the assumption that each individual has his or her opinion on the trustworthiness of another. It is believed that these opinions can be collected, exchanged and evaluated. The initial application areas for this distributed trust model were: a reputation guided searching system and a recommendation system for security purposes. The first problem with the Poblano trust model is that the equations for calculating/updating trust are simple and arbitrary. Thus, the accuracy and general applicability of this formula is questionable. The second problem with the model is that it seems specifically designed for solving the problem of searching distributed networks and the validation of the search results and sources. Finally, the Poblano framework appears to have no facilities for the specification of constraints. This severely limits its usability. Thus, Poblano may not be easily adapted to work in applications domains that are dissimilar to this one.

The third problem is that it may suffer from the same key problems faced by public key certificates.

Some other popular systems policy based trust management systems are:

RT [160], which stands for Role-based Trust Management Framework is a successor to TrustBuilder. RT introduces the notion of attribute acknowledgement policies, which protects information about the possession of credentials. It also adds notions of a trust target graph protocol and the distributed storage of credentials. In [161], an Active Trust Management System for Autonomous Adaptive Survivable Systems is proposed. Kagal et al. [162] described a trust management framework that uses a system of rights and delegations, as well as digital certificates, to facilitate trust management. Their architecture assumes that each group of agents is protected by security agents, which are responsible for authorizing access to services/resources within the group. A Hybrid Trust Management Model For MAS (Multi Agent Systems) Based Trading Society is proposed in [163]. In this section, the various views of trust were discussed and their common traits identified. A common flaw with all these solutions is that they are used to identify a static form of trust, usually at the discretion of the application coder (that is, the programmer inserts code to evaluate trust, often at the start of a session). However, trust can change over time. Typically, a customer uses an unknown service provider with some trepidation but if the service provided is high quality over a period of time, the customer's trust in the service provider increases. In order to handle this dynamic property of trust, solutions should design a framework that assumes non-monotonicity. They should be able to adapt to the changing conditions of the environment in which the trust decision was made. Systems should be able to incorporate their own experiences (or that of others) in their decision-making process. Systems change and evolve so there is a need to monitor trust relationships to

determine whether the criteria on which they are based still apply. This could also involve the process of keeping track of the activities of the trustee and of determining the necessary action needed when the trustee violates the trustor's trust. After an extensive survey on popular Policy based trust management system in this section, the next section explores Reputation based trust management systems.

Reputation Based Trust Management In Reputation based trust management system, the focus is on trust computation models capable of estimating the degree of trust that can be invested in a certain party based on the history of its past behavior. It is intended for distributed systems where a system only has a limited view of the information in the whole environment. New trust relationships and trust values are inferred based on the available information. On a decentralized network like Grid, users can see from where the information arrives, as well communicate their opinions on both the information they have acquired and the peers who are its source. These personal opinions can be collected, exchanged and evaluated. Furthermore, these opinions, when evaluated, can be used as guidelines for searching for information and recommending information sources, thus, creating decentralized, personalized "web-of-trust". Trust becomes a social contract with social implications for the participants. Each user of such decentralized setup will develop a reputation among his peers. One of the first applications to handle reputations in an automated fashion on a genuinely large scale was the "web of trust" system introduced in Phil Zimmermann's Pretty Good Privacy (PGP) [164]. This was the first program to bring public key cryptography to the masses. In public key cryptography, there are two keys per user. One is public and can be used only to encrypt messages. The other key is private and can be used only to decrypt messages. A user publishes his public key and keeps the private key safe. Then others can use the public key to send him messages that only he can read. Some

of the common reputation based trust management models for Grid Computing are discussed here:

Aberer and Despotovic proposed P-Grid [165] in which reputation is an assessment of the probability that an agent will cheat. To compute reputation they have used data analysis of former transactions. Their trust is binary; an agent can perform a transaction correctly or not. P-Grid considers that usual trust exists and therefore, disseminate only dishonest information as relevant. P-Grid names this information as complaints. Therefore, agent p after detecting the malicious behaviour of agent q will store a complaint $c\{p, q\}$. The method is quite suited for Peer-to-Peer (P2P) approaches and also for decentralized desktop Grids. With regard to a standard Grid, anyway, the usability of the method is under question, as it does not address the QoS and SLAs and also not the virtual organization formation.

Sepandar et al. proposed Eigentrust [166] in which some important issues were proposed as important for peer to peer reputation systems. The first being self-policing into which no central authority should exist and the peers should enforce the ethical behaviour by themselves. The second is anonymity where peer reputation should be associated with an opaque identifier. Then, the system should not assign profit to newcomers, should have minimal overhead and be robust to malicious collectives of peers.

Abrams et al. [167] proposed strategy proof partition mechanism that provides incentives for peers to share files, is non-manipulable by selfish interests and approximates trust scores based on EigenTrust. The basic idea behind the partition mechanism is that the peers are partitioned into peer groups and incentives are structured so that a peer only downloads from peers in one other peer group. In addition to theoretically guaranteeing non-manipulability, in practice their trust system performs nearly as well

as EigenTrust and has better load-balancing properties.

Doing some extensive experiments, EigenTrust showed a good performance of the trust model in the peer-to-peer setup but not in Grid. Although, they could not totally reduced the failure rate in the system, but the improvements were significant.

XenoTrust [168], [169] is another trust and reputation management architecture. It consists of three main components: XenoServer, XenoCorp and XenoServer Information Services (XIS). XenoServers provide services to the client. XenoCorp provides authentication, auditing, charging and payment services. XIS is used for storing the XenoServer status updates. The XenoTrust architecture introduces two levels approach of managing trust, called authoritative and reputation based. Authoritative trust is a boolean property established between a XenoCorp and the clients and servers that register with it. Reputation-based trust, on the other hand, is a discrete continuous property which quantifies, in a particular setting, the trustworthiness that one component ascribes to another. It is distributed and highly subjective, in the sense that each entity has its own, independent view of other's reputations.

PeerTrust [170] was developed at Georgia Tech with peer-to-peer based electronic applications in mind. The PeerTrust metrics are derived from a combination of parameters like feedback, total number of transactions, factor of credibility, transaction context and community context. PeerTrust does not use a centralized database for storing the trust information. Rather, the trust information is stored in a distributed manner over the network. Each peer or a node in the network has a trust manager that is responsible for feedback submission and trust evaluation. There is a small database that stores a portion of the global trust data and a data locator for placement and location of trust data over the network. The trust computation is based on the computation of trust data collected from different nodes or peers.

SeGO [171] [172] system was developed at the University of Southern California developed for secure scheduling of a large number of autonomous and indivisible jobs to Grid. A unique feature of the work is that the authors use a fuzzy inference approach to binding Security in trusted Grid computing environment. SeGO scheme does not explicitly deal with the storage of trust information.

GridEigenTrust [173] exploits the beneficial properties of EigenTrust [166], extending the model to allow its usage in Grids. GridEigenTrust integrates the trust management system as part of the QoS (Quality of Service) management framework, proposing to probabilistically pre-select the resources based on their likelihood to deliver the requested capability and capacity. The basic framework of EigenTrust was taken and adapted for Grid requirements, resulting the GridEigenTrust model. GridEigenTrust proposed to integrate trust in a QoS management, trust should be related to multiple existing contexts. In a Grid, there is a need to address entities, organizations and virtual organizations. The approach used in GridEigenTrust [173] is one of the few from literature to propose a reputation service as a way to improve Quality of Service (QoS) management in Grids. Although it presents the design of the system, but experiments have not been presented in order to prove the efficiency of the approach.

PathTrust [174], was proposed as a reputation system for member selection in the formation phase of a virtual organization. In PathTrust, to enter the Virtual Organization (VO) formation process, a member must register with an enterprise network (EN) infrastructure by presenting some credentials. Besides user management, EN supplies with a centralized reputation service. At the dissolution of the VO, each member leaves feedback ratings to the reputation server for other members with whom they experienced transactions. The feedback ratings can be positive or negative ratings. PathTrust arranges the participants in a graph structure similar with the one of

NICE [175]. Each edge in the graph is weighted with the trust between the nodes at the ends of the edge. As in NICE [175], for assessing the reputation between two nodes in the graph, the PathTrust algorithm selects the path with the maximum weight. Like in the EigenTrust [166] approach, the trust value is seen as the probability of selecting a participant from the list of possible alternatives. When PathTrust scheme is evaluated against the EigenTrust algorithm and against attacks by reporting fake transactions in the system, it seems that with EigenTrust, a cheater can gain more profit than with PathTrust. However, in case of selection of random participants, EigenTrust loses its advantage once cheating was introduced in the system. This loss occurs also with PathTrust, but is much lower. Therefore, to prevent cheating, the authors propose the usage of a transaction fee. PathTrust is one of the first attempts to apply reputation methods to Grids by approaching VO management phases. They approached only partner selection and did not tackle organizational aspects. Their model still lacks the dynamics, as the feedback is collected only at the dissolution of the VO.

NICE [175] is a platform for implementing cooperative applications over the Internet. NICE uses a distributed trust evaluation mechanism. Whenever there is a transaction between two different nodes or peers, the peer receiving the service signs a cookie showing that the peer generating that service has successfully completed the transaction. Therefore, each node maintains a trust relationship and trust value based on the transaction it has received. The node generating transaction can store the signed cookies to prove its trustworthiness sometime later.

Abdul-Rahman and Hailes [176] proposed a model for computing the trust for an agent in a specific context based on the experience and recommendations. The trust values were kept discrete: very trustworthy, trustworthy, untrustworthy and very untrustworthy. Each agent stores the trust values for the agents interacted, therefore, the trust

model is distributed. Each agent also stores the recommender trust with respect to another agent. The recommender trust value are semantic distances applied for adjusting the recommendation in order to obtain a trust value. They proposed a method for evaluating and combining recommendations and updating the trust value. As the model is based on the set theory, each agent has to store all history of past experiences and received recommendations. On a system with a lot of participants and frequent transactions, each agent should have a large storage with this respect. The network traffic is caused by the messages exchanged between agents in order to get reputation information.

TrustMe [177] is another approach for decentralized and unstructured P2P networks. It is a user-based approach, adopting the principle of obtaining references about a peer, before engaging in a transaction with that peer.

Advogato [178] provided a trust metric, called Advogato maximum flow trust metric, in order to discover which users are trusted by members of an online community and which are not. Hereby, trust is computed by a centralized community server and considered relative to a seed of users enjoying supreme trust. However, the metric is not only applicable to community servers, but also arbitrary agents which may compute own trusted peers and not for the whole community they belong to. In this case, the agent itself constitutes the singleton trust seed. This model does not provide support for weighted trust relationships.

Appleseed [179] can be seen as an improvement over Advogato model in terms of computation of local trust metric and its propagation. It also has working similarity with Google's PageRank [180] is a ranking algorithm whose original goal is to sort backlinks so if there were a large number of backlinks for a document, the best backlinks could be displayed first.

The Josang et al.'s [181] subjective logic is a trivalent one, an opinion could have 3 degrees of values: belief (b), disbelief (d) and uncertainty (u), with $b+d+u=1$ with $\{b, d, u\}$ [182]. The main contribution of Josang subjective logic is a clear representation of the logic each node in the network should possess in order to manage the experience and the received referrals. Josang shows the compatibility between the subjective logic and the PGP authentication system, demonstrating the usage of the trust values in Grid-like networked environments (based on layered certifications). Josang et al. [181] applied this reputation mechanism for improving service discovery in a P2P environment, combining the reputation computation with distributed hash-table routing structure. In this development reputation is built only based on experience and received feedback.

Sabater and Sierra [183], proposed a model, named REGRET that considers the following dimensions of the reputation: the individual dimension: which is the direct trust obtained by previous experience with another agent, the social dimension which refers to the trust of an agent in relation with a group and the ontological dimension which reflects the subjective particularities of an individual. REGRET focuses on SLAs between two parties, with several variables under interest. The service composition is a well-desired property of Grid systems. Besides the internal agent database, another database is required at the level of a group in order to store the group-based reputation. This is a mean of centralization. Overall, the REGRET system could be of interest for Grids as it allows a way of aggregating the reputation at the level of a group and approach the service composition, which is a particularity of Grid systems.

Farag and Maheshwaran [184], [185] proposed and extended a trust model that deals with behavior trust, how it is evolved based on transactions between entities and how it is managed in Grid computing systems.

Ramchurn [186] defined a system on the terms of a contract (Service Level Agreement - SLA). The model principles do not differ too much from the one of Sabater and Sierra [183], but it has the advantage of entering more in the details of the establishment of a SLA. It can be worth for the Grid community, as the authors show how trust can be incorporated in the SLA bilateral negotiation process by permitting the adjustment of the proposed values for the issues of a contract in a more reliable way. Even more, when the trust that the negotiation partner will supply the agreed values in the contract is low leading to negative utility expectations, the model shows how an agent can further require more issues in the contract (as a new quality certification) in order to secure a positive utility expectation. There are other trust management systems such as Pervasive Trust Management (PTM) [187] used for dynamic open environments and Personalised Trust Model (PET) [188] with Reputation and Risk Evaluation for P2P Resource sharing. Silaghi has reported Reputation-based trust management systems and their applicability to Grids in [189].

Reputation systems have been widely used for Internet sites which mainly uses summation-based reputation systems. These systems are based on counting all feedback values received by an entity. The feedback values in the form of votes can be simply counted on the behalf of the user or they can be averaged or weighted. As summation-based reputation systems are mainly used in e-commerce marketplaces, they are mostly centralized. The advantage lies in simplicity of the reputation scheme. This makes the reputation value to be easily understood by the participants and allows a direct conversion between reputation assessment and trust. The most widely known reputation system of this kind is eBay [190]. In eBay, the feedback can be submitted as -1 to 1. Other systems for ecommerce applications are Amazon, Epinions, BizRate etc. [181]. In the Amazon bookstore, reputation is assigned to books and to reviewers.

Regarding the books, the reputation of a book is the average score the book received from its reviewers. A reviewer can assign to a book between 1 to 5 stars. Each reviewer has its own reputation. Each time a review is considered helpful by a user, the reviewer receives a vote. The reviewers are ranked based on the votes they received from users. Similar with eBay [181], the Amazon reputation system is a centralized one, the reputation is a common knowledge and it has the advantage of simplicity. Anyway, in Amazon, the reputation does not have such a great impact on the whole marketplace, as it can only affect the buying decision, not the price at which the transaction happens. It is also expected that reviewers to receive positive feedback, but, unlike in eBay, Amazon does not display how many negative votes a reviewers received. Amazon reputation system is not a transactional one, as one can vote a reviewer even without buying the item under review.

The kind of reputation systems for e-commerce applications are not too valuable for Grids, as they do not allow reputation to directly influence the transaction execution in the system.

Dingledine et al. [191] have proposed the following set of basic criteria for judging the quality and soundness of reputation computation engines:

- (i) Accuracy for long-term performance: The system must reflect the confidence of a given score. It must also have the capability to distinguish between a new entity of unknown quality and an entity with poor long-term performance.
- (ii) Weighing current behaviour: The system must recognise and reflect recent trends in entity performance. For example, an entity that has behaved well for a long time but suddenly goes downhill should be quickly recognised as untrustworthy.

- (iii) Robustness against attacks: The system should resist attempts of entities to manipulate reputation scores.
- (iv) Smoothness: Adding any single rating should not influence the score significantly.

The criteria (i), (ii) and (iv) are easily satisfied by most reputation engines except for the most primitive such as taking a rating score as the sum of positive minus negative ratings such as in eBay's feedback forum. Criterion (iii) on the other hand will probably never be solved completely because there will always be new and unforeseen attacks for which solutions will have to be found. Such security threats in distributed systems have been covered extensively by Marmol [192], [193]. After a comprehensive literature survey, following parameters have been derived as essential to be catered by any trust management system.

Identity trust: Identity trust is the trust that a subject holds on other entity without making any intermediate entity. It focuses on verifying the authenticity of an entity and determining the authorizations that the entity is entitled to access and is based on techniques including encryption, data hiding, digital signatures, authentication protocols and access control mechanisms.

Behavior trust: Behavior trust may be considered as reputation of an entity as seen by other entities. Participants behave either "good" or "bad". In most cases, "good" behavior reflects the expectations of a participant to simply receive a response from another participant involved in an interaction or sometimes to get accurate results. If an interaction party behaves differently from these "normal" expectations, its behavior is labeled as "bad". Participants with "good" behavior are considered as trusted ones and are thus eligible for future interactions. Participants with "bad" behavior have only minor or no possibility to be considered for further interactions.

Reputation: The reputation of an entity is an expectation of its behavior based on

other entities observation or information about the entity's past behavior within a specific context of a given time. Reputation of an entity is calculated by the feedback information submitted by the interacting player which consists of trustworthiness and recommendation information as well.

Feedback: A piece of information about the services provided by any entity that can be discrete i.e trustworthy, not trustworthy etc., or continuous or binary.

Trust and Reputation Updation: Trust and Reputation information is updated a successful transaction.

Trust and Reputation Decay: Trust and reputation values decays with time. The positive or negative effect of a successful interaction reduces over the time.

False Recommendation: Reputation of an entity is calculated from recommendation of several other entities. Any entity can give fraudulent recommendation, which is taken care of by aggregating all the recommendation given by a particular entity. Audit trails enable finding out false recommendation given to a domain/entity to garble the values.

Trust Inheritance: In a Grid environment, entities are dynamic, that is, they can join and leave a virtual organization at any time. Hence, a trust model for such an environment should have mechanisms for managing trust for such entities. Trust values can be inherited by the new users when becoming part of Grid.

Trust level: Trust can be categorized into levels. Trust can range from ht to hu as shown in Table 2.1 where et is rarely achievable therefore, not considered for real models.

Risk assessment: Risk assessment accounts for the bad behavior of the entity. Even a good known entity may also behave badly in some situations. Risk assessment checks the risk involved in transacting with a particular entity.

Table 2.1: Trust Level

Trust level	Significance
et	Extremely high trustworthy
ht	Highly trustworthy
t	Trustworthy
u	Untrustworthy
hu	Highly untrustworthy

Transitive Recommendation: If A trusts B and B trusts C then A may also trust C. This is called transitivity. In open distributed environments trust information can be collected from recommenders and trust bases to check the authenticity of the users and then transaction may be committed.

Evolving trust as a newcomer: As a new entity joins a domain, it initially has no Trustworthiness and reputation values. A newcomer is allowed to interact with domain having certain threshold limit for services and it can then build its own trust values.

Weightage of identity trust and reputation: An identity for which both direct trust and recommendation trust exist, direct trust is given a higher weightage than the recommendation trust.

Recommendation Aggregation: All the recommendation given for a particular entity is aggregated to check any malicious or fraudulent information.

Trust Symmetry: If Alice trust Bob then does Bob trust Alice? The answer is not necessarily yes! This situation is called trust asymmetry problem.

Trust relationships: Determining trust relationship is essential while accessing resources/services. Trust Relationships can be direct, half, full, partial, indirect and recommended depending upon the nature of the transaction and has been explained below.

- Direct Trust : Trust that an entity has on a user/service/domain without any intermediate user/service/domain or other entity.
- Full Trust: Trust that an entity has on a user/service/domain if it trust on all the services provided by that user/service/domain.
- Partial Trust: Trust that an entity has on a user/service/domain if it trusts on some of the services provided by that user/service/domain.
- Recommended Trust: Trust that an entity has on user/service/domain that is recommended by other entities.
- Privacy Trust: Trust that an entity has on user/service/domain on the privacy features provided by other entity.
- Authentication Trust: Trust that an entity has on user/service/domain on the authenticity of an identity certificate signed by a certificate authority

Trust context factor: At any time t , the purpose of using services of a service provider by a service requestor can be termed as the trust context factor, which may vary from application purpose. The trust weights may be assigned and evaluated likewise.

Historical accumulation of past behavior: An entity that interacts with another entity, after evaluation of trust, has to store the value for future use.

Classification of services: The services can be classified based on the confidentiality and ease of use parameters.

Trust as a subjective quantity: Trust is taken as a subjective quantity which can be termed as belief in services or an entity to perform as promised, earlier.

Trust Threshold: It is taken as a minimum value required depending on the sensi-

Table 2.2: Responses to Comparison in Tables 2.3 and 2.4

Type of Response	Value	Description
Values	Yes	Feature exists
	NE	Feature does not exist
	C	Feature considered
Feedback	Bin	Binary
	Cont	Continuous
	Dis	Discrete
	NA	Not Available

tivity of the application, service requested or provided to establish trust relationship with any entity.

Interoperability: Interoperability aims at federating the use of user’s credentials and providing a unique authentication and authorization framework. Users logged into disparate middleware are allowed to log in to the system and access Grid.

Trust View: The trust view depends on three dimensional view having components of Belief, Disbelief and Plausibility [182].

Policy based systems dealt mostly with identity trust and left less scope of comparison on the parameters discussed here. After a comprehensive literature survey, the popular Reputation based trust models are compared in Table 2.3 and 2.4 in terms of their support to various parameters as discussed. Table 2.2 shows the responses description to the comparison in Table 2.3 and 2.4.

Table 2.4: Trust Management Systems in various Computing Paradigms (continued)

Features/Trust Management Systems	Farag and Muthucammaro	Abdul Rahman and Hailes	Ching li	TrustMe	PET05	Grid Eigen-Trust	PathTrust	XenoTrust	SeGo	Ramchurn
Identity Trust	Yes	Yes	Yes	Yes	NE	NE	Yes	Yes	Yes	NE
Behavior Trust	NE	C	NE	NE	C	Yes	Yes	NE	NE	Yes
Reputation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Type of Feedback	Dis	Dis	Dis	NE	Dis	Dis	Dis	Dis	Cont	Dis
Trust and Reputation Update	Yes	Yes	Yes	NE	NE	Yes	Yes	NE	Yes	Yes
Trust Decay	Yes	Yes	NE	NE	Yes	Yes	Yes	NE	NE	NE
Reputation Decay	Yes	Yes	NE	NE	NE	Yes	Yes	NE	NE	NE
False Recommendation	Yes	NE	Yes	NE	NE	Yes	NE	Yes	NE	Yes
Trust Asymmetry	Yes	Yes	Yes	NE	NE	NE	NE	NE	NE	NE
Trust Inheritance	C	NE	Yes	NE	NE	NE	NE	NE	NE	NE
Trust Level	C	Yes	Yes	NE	NE	NE	NE	NE	NE	NE
Risk Assessment	NE	NE	NE	NE	Yes	NE	Yes	NE	NE	NE
Transitive Recommendation	NE	C	Yes	NE	NE	NE	NE	NE	NE	NE
Evolving trust as a newcomer	C	C	NE	NE	NE	NE	Yes	NE	NE	NE
Weightage Identity Trust and Behaviour Trust	C	Yes	NE	NE	NE	Yes	NE	NE	NE	NE
Recommendation Aggregation	Yes	Yes	NE	NE	Yes	Yes	Yes	NE	NE	Yes
Quality of Service	NE	NE	NE	Yes	NE	Yes	NE	NE	NE	NE
Trust Relationships	NE	NE	NE	NE	NE	NE	Yes	NE	NE	NE
Service Level Agreement	C	NE	NE	NE	NE	Yes	NE	NE	NE	Yes
Trust Context Factor	Yes	Yes	NE	NE	Yes	NE	Yes	NE	NE	NE
Historical accumulation of past behavior	C	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Classification of Services	NE	NE	NE	NE	NE	NE	Yes	Yes	Yes	NE
Trust as subjective quantity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Trust Threshold	Yes	Yes	Yes	Yes	Yes	NE	Yes	Yes	Yes	Yes
Interoperability	NE	NE	NE	NE	NE	NE	NE	NE	NE	NE
Trust View	1-D	1-D	1-D	3-D	1-D	1-D	1-D	1-D	1-D	1-D

In Table 2.3 the following trust management systems are compared: (1) ebay [190], (2) Amazon, (3) Epinions, (4) Advogato [178], (5) REGRET [183], (6) EigenTrust [166], (7) PeerTrust [170], (8) P-Grid [165], (9) NICE [175] and (10) Josang [181].

In Table 2.4 the following trust management systems are compared: (11) Farag and Muthucamaro [185] (12) Abdul Rahman and Hailes [176], (13) Ching li [194], (14) TrustMe [177], (15) PET05 [188], (16) GridEigenTrust [173], (17) PathTrust [174], (18) XenoTrust [168], (19) SeGo [171] and (20) Ramchurn [186].

Inconsistencies in current trust relationships highlight the need for a flexible, general - purpose trust management system that can navigate complex trust domains in a Grid. A trust management system for large scale distributed systems like Grid and web should be scalable, reliable and secure. There was an observation that although all the models catered Grid Computing and Distributed Computing infrastructure, however only few models incorporated Service Level Agreements and Quality of Service parameters. Only one model provided Three dimensional view of trust. Table 2.3 and 2.4 showed that there is no reputation based trust model which addresses all the features deemed essential in a Grid environment as per the review conducted. All the trust management models supported identity trust except one. Behaviour trust was also considered in some models. The type of feedback also varied for all the models as mentioned using Table 2.2 except one model which was not having any feedback collection. Trust and Reputation Updation was still considered for all the models. The trust decay, reputation decay, false recommendation, transitive recommendation, evolving trust as a newcomer, trust relationships, trust asymmetry and interoperability was rarely considered in designing and functioning of various models. However, trust inheritance, trust level, risk assessment, recommendation aggregation, historical accumulation of past behaviour and trust as a subjective quantity was still considered by

most of the models. Very few parameters are supported by the trust management models, as discussed and compared here. Therefore, leading to development of a flexible trust management model for complex and dynamic Grid Computing environments.

2.3 Interoperability and Trust Management Issues

Security requires the three fundamental services: authentication, authorization and encryption. A Grid resource must be authenticated before any checks can be done as to whether or not any requested access or operation is allowed within the Grid. Once the Grid resources have been authenticated, the Grid user can be granted certain rights to access a Grid resource. This, however, does not prevent data in transit between the Grid resources from being captured, spoofed or altered. The next section presents and analyzes interoperability and trust management issues such as authentication and authorization in Grid Computing Environments.

2.3.1 Authentication

Authentication is the foremost Security requirement to be catered in a Grid Computing scenario. It deals with verifying the identity of a user or resource. This identity can be attributed to the user of the system, but in a Grid environment, the identity can be attributed to any component accessing any other component. Authentication is achieved through the presentation of some Security credentials that cannot be forged. In a Grid environment, authentication is critical not only for users that are authenticating when they log in to the Grid system, but also for the resources that joins the Grid environment.

2.3.1.1 Authentication in existing middleware:

A number of important projects address the challenges identified in the previous section related to authentication. Major among them include Microsoft .NET Passport system [195], Entrust TruePass portfolio [196], MyProxy [197], Kerberized Certificate Authority (KCA) [198], [199], Password Safe [200], Microsoft Trust Bridge [201], Liberty Alliance Project [202], CredEx [203] etc. The goal of Microsoft .NET Passport system [195] is to provide a uniform and distributed network to perform authentication. Passport [195] implements single sign-on solution for the web by providing users one set of credentials (id, key, etc.) which are centrally stored. To use single sign-on, the target service must be Passport enabled. A user can carry a single Passport credential among several web applications that support the Passport scheme. In this manner, the user needs only a single login name and password in order to access disparate applications. The user authenticates to Passport which in turn provides a cookie and the user presents this cookie to the web application. Thus, it functions much like Kerberos tickets [204]. It does not support proxy certificates. The Entrust TruePass portfolio [196] is another system that promises convenient web based authentication using PKI based Entrust Digital IDs and signatures. It also requires the target Web Services to be TruePass enabled. Entrust TruePass is an enhanced Web Security solution that leverages the advanced public-key infrastructure (PKI) capabilities provided by the Entrust Authority Security Manager. It enables the protection of web site resources and applications and adds accountability and privacy to online transactions [196]. But in this credential management and delegation features are not supported. MyProxy [197] on the other hand is a true credential repository and supports the storage of multiple user credentials. It is developed to meet the credential management requirements of the Grid community. The MyProxy system is the implementation of the Virtual Soft

Token system proposed in [205], where the X.509 proxy certificates are used to store and retrieve user credentials without having to expose the private key. MyProxy is being extended with a name change to GridLogon to reflect its enhanced capabilities. The main function of the GridLogon service is to issue X.509 credentials to clients that are used by the client to securely access Globus Toolkit enabled Grid resources [206]. KX.509 1.0, from the University of Michigan, is a Kerberized client-side program that acquires an X.509 certificate, using existing Kerberos tickets. The Globus Toolkit [71] normally uses X.509 credentials [207]. KX.509 allows a user to authenticate to a host that is running Globus software using Kerberos tickets instead of requiring X.509 certificates to be installed [197]. X.509/KCA [199] is a system that just supports the exchange of Kerberos ticket-granting ticket for X.509 proxy certificate but it is not a credential repository. Another system Password Safe [198] is an example of a password database that allows users to store passwords for different accounts. Password Safe protects passwords with the Blowfish encryption algorithm [200]. The problem with Password Safe is that it involves the exchange of a single type of token only. It is not a single sign-on and delegation solution. The Liberty Alliance Project [202] is an effort, initiated by Sun Microsystems and now involving many companies, with the goal of facilitating authentication on the web for both human users and automated mechanisms. Liberty Alliance Project defines how various identity providers (which store information related to user identities) can be linked together to support single sign-on across multiple distinct services. The framework provides standards for sharing attribute information across a trusted set of entities called the Circle of Trust (COT). The user is authenticated by the identity provider and after authentication, user can access services provided by service providers. The sessions are transferred between different service providers. The user may possess different identities with different service

providers as long as they are connected through the COT constraint. The weakness of Liberty Alliance Project is that it falls short of addressing support for diverse authentication token types [203]. Microsoft's TrustBridge [201] is federated identity effort that builds on WS-* set of Web Services Security specifications like WS-Security, WS-Trust, WS-Federation etc. TrustBridge enable businesses to share user identity information between applications and organizations. It allows different organizations using the Windows operating system to exchange user identities and interoperate in heterogeneous environments using industry standard XML Web Services protocols. CredEx [203] is an open source standards based Web Service that facilitates the secure storage of credentials and enables the dynamic exchange of different credential types using WS-Trust token exchange protocol. With CredEx, a user can achieve single sign-on by acquiring a single credential but support for credential delegation is not there. MyProxy [197] and CredEx [203] are indeed a key inspiration for the proposed authentication framework. More attempts in this area also exist like [208], [209], [210] and [211] but these do not support all types of authentication features required in Grid systems.

After authentication, authorization is the next important Security requirement discussed in the next subsection.

2.3.2 Authorization

Authorization, in simple terms, deals with issues like who can access what services/resources under what conditions. Grids present several unique authorization related challenges. There are several factors that make authorization hard in Grid systems such as:

- (i) Users and resource pool is large and dynamic
- (ii) Resources have different authentication, trust and authorization requirements

- (iii) Computations span over multiple domains
- (iv) Users have different roles/privileges in different domains

All these factors make authorization a big and challenging issue.

2.3.2.1 Authorization in existing middleware:

A lot of projects like Akenti [212], CAS [213], PERMIS [78], CARDEA [214], PRIMA [215], GridShib [83], [81], GUMS [216], SESAME [217] and VOMS [131] are developed to address authorization related issues in one or other form in distributed environments but they have their limitations also. Central Authorization Service (CAS) and Virtual Organization Membership Service (VOMS) are VO level authorization systems whereas Akenti and PERMIS are resource level authorization systems. GUMS is a Grid user management system. Following paragraphs briefly discuss each of these.

Akenti [212] is an access control mechanism that uses digitally-signed certificates to define and enforce an access policy for a set of distributed resources that have multiple, independent and geographically dispersed stakeholders. The stakeholders assert their access requirements in use-condition certificates. Users are identified by X.509 identity certificates. During a request to use a resource, a policy engine collects all the relevant certificates and decides if the user satisfies all the requirements [212]. Akenti allow different stakeholders to express policies specifying how resources can be accessed but the policy language is different from XACML. Akenti policy is expressed in XML and stored in three types of signed certificates: policy certificates, use-condition certificates and attribute certificates. Thus, Akenti policy language model is different from XACML policy language model.

CAS [213] is an approach to the representation, maintenance and enforcement of policies and provides a scalable mechanism for specifying and enforcing these policies. This

approach allows resource providers to delegate some of the authority for maintaining fine-grained access control policies to communities, while still maintaining ultimate control over their resources. The owners of resources grant access to a community account as a whole. The CAS server is responsible for managing the policies that govern access to a community's resources. It maintains fine-grained access control information and grants restricted GSI proxy certificates to the users of community [219]. CAS can also be used for managing role-based sub groups of a virtual organization [218]. CAS also support mechanism to delegate permissions on a set of resources distributed across different administrative domains but it focuses on centralized specification of community policies governing access to collection of resources. Moreover, in CAS, policies are not expressed using XACML.

PERMIS [78] is a policy driven RBAC Privilege Management Infrastructure (PMI). PERMIS implements Role Based Access Control (RBAC) scheme in which rights are associated with roles rather than with specific entities. PERMIS describes a policy driven role based access control system. The user's roles and the policy are stored in X.509 Attribute Certificates. The policy, written in XML, describes who is trusted to allocate roles to users and what permissions each role has. Access control decisions are made by an Access Control Decision Function. The decision is made according to the requested mode of access, the user's trusted roles and the policy [78]. In PERMIS, policies are written in XML.

Cardea [214] is a distributed authorization system that facilitates dynamic access control. It is developed as part of the NASA Information Power Grid [220], which dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and requester rather than considering specific local identities. Potentially accessed resources within an administrative domain are protected by local access con-

trol policies, specified with the XACML [75] syntax, in terms of requester and resource characteristics. The exact information needed to complete an authorization decision is assessed and collected during the decision process itself. This information is assembled appropriately and presented to the Policy Decision Point that returns the final authorization decision for the actual access request together with any relevant details. Cardea is currently implemented in the Java language. In Cardea, policies are defined with respect to high level identities such as entity's distinguished name. The authorization decisions depend heavily on the attributes a service requester holds.

PRIMA [215] focuses on the issues of management and enforcement of fine-grained privileges. PRIMA mechanisms enable the use of fine grained access rights and adhoc and dynamic collaboration scenarios. PRIMA provides tools for end users and administrators to manage privileges for the resources they are authoritative for through X.509 Attribute Certificates that carry privilege and policy statements. An access control decision function in the form of an authorization module for the Globus Toolkit authorizes requests based on the combination of user access privileges with resource policies and provisions low-level access control enforcement functions with decision qualifications [215]. PRIMA system is particularly motivated by the desire to support spontaneous, short lived collaborations among small group of Grid users.

GridShib [83], [81] is National Science Foundation (NSF) funded project between National Center of SuperComputing Applications (NCSA) and the University of Chicago. The goal of the project is to integrate GSI [207] and Shibboleth [80] to provide the needed capabilities for a robust attribute infrastructure. This project will deliver a framework that allows participants in multi-organizational collaborations to control the attribute information that they want to publish, share and reveal to other parties. Those parties will also be able to determine whether they possess the capabilities to

access a service by matching their capabilities with the service's shared policy of required attributes. The pseudonymous interactions will be supported through the use of anonymous public key credentials that are mapped to the client's identity at the client's own discretion [206].

GUMS [216] is another system that automates user registration and management for a local Grid site. The model allows users to register once with a VO and provide all the computing sites the information they need with the required level of trust. Tools have been developed to allow sites to automate the management of local accounts and the mapping between Grid identities and local accounts [216]. GUMS is not an authorization solution instead it is a user management system.

SESAME [217] is dynamic context-aware access control mechanism for pervasive applications. SESAME complements current authorization mechanisms to dynamically grant and adapt permissions to users based on their current context [217]. The proposed mechanism extends the Role Based Access Control (RBAC) model while retaining its advantages. The model dynamically adjusts Role Assignments and Permission Assignments based on context information. Each user is assigned a role subset (by the authority service) from the entire role set. Similarly the resource has permission subsets for each role that will access the resource [217]. SESAME does not make use of any policy language like XACML to express access control policies.

Virtual Organization Membership Service (VOMS) [131] manages authorization information in Virtual Organization scope. The VOMS System consists of four main components namely User Server, User Client, Administration Client and Administration Server. User Server receives requests from a client and returns information about the user. User Client contacts the server presenting a user's certificate and obtains a list of groups, roles and capabilities of the user. Administration Client is used by the

VO administrators for adding users, creating new groups, changing roles etc. Administration Server accepts the requests from the clients and updates the database. The VOMS architecture uses the authentication and delegation mechanisms provided by the Globus Toolkit Grid Security Infrastructure (GSI) [207]. VOMS [131] constitutes a system conceptually similar to CAS. It has a community centric attribute server that issues authorization attributes to members of the community. A universal, scalable authorization and access control framework called RB-GACA for Grid systems has been proposed in [219] but the framework does not provide facilities to integrate different authorization mechanisms. It is based on role based access control mechanism.

This section presented an analysis of interoperability and trust management issues such as authentication and authorization. The comparative study of these projects and solutions existing in Table 2.5 in terms of their support for authentication, authorization, interoperability and trust management showed that there were hardly any projects available for supporting interoperability and trust management. Although authentication and authorization were supported by most of the projects and middleware. It presented interoperability and trust management in security solutions as a research direction and initiative to be undertaken. The next section discusses Security in various popular Grid middleware.

2.4 Security in Grid Middleware

Grid can be visualized as an infrastructure of varied resources connected together by a Grid middleware, which sits on top of this infrastructure so as to enable collaboration between them. Grid Middleware is responsible for access to a Grid resource, for communication with the user, for scheduling, error handling, data management, job

Table 2.5: Comparison of Middleware and Projects in terms of Security

Middleware / Projects	Authentication	Authorization	Trust	Interoperability
CredEx	Yes	No	No	No
Entrust TruePass	Yes	No	No	No
Microsoft .NET Passport system	Yes	No	No	No
Kerberized Certificate Authority	Yes	No	No	No
Microsoft Trust-Bridge	Yes	No	No	No
MyProxy	Yes	No	No	No
Password Safe	Yes	No	No	No
EPAL	No	No	No	No
P3P	No	No	No	No
Akenti	No	Yes	No	No
CARDEA	No	Yes	No	No
CAS	No	Yes	No	No
PERMIS	No	Yes	No	No
PRIMA	No	Yes	No	No
SESAME	No	Yes	No	No
VOMS	No	Yes	No	No
Shibboleth	Yes	Yes	No	No
GridShib	Yes	Yes	No	No

management and above all, for enabling Security. Thus, middleware is most important part of Grid, as it manages all the communication and hides the difference between all entities connected together in a Grid. Different research groups and communities have developed their own middleware based on their usage and requirement. It is pertinent to mention here, these middleware are incompatible with each other and each of them uses their own components, interfaces and protocols which results in slightly dissimilar functionalities to the user. An end user is then occupied with functional details of middleware components instead of concentrating on application level details.

In this section, an overview of techniques incorporated for adopting Security in existing middleware has been discussed. The taxonomy followed is based on the Security architecture of GLOBUS Toolkit, UNICORE, SUN N1GE6 (SUN Grid Engine), CONDOR and Legion. Based on this taxonomy an analysis of Security in Grid middleware has also been done.

2.4.1 GLOBUS TOOLKIT

The Globus Alliance produces widely-used Grid middleware known as the Globus Toolkit [71]. The Globus Toolkit, now at version 5, consists of a collection of services which can be deployed to provide a working Grid. These consist of information services, data management services and computing resource services. The information services keep track of Grid metadata for the purpose of monitoring and allowing discovery of data and computation resources. Versions 1 and 2 of the Monitoring and Discovery Service (MDS) are based on Lightweight Directory Access Protocol (LDAP) [221]. Resource information from individual Information Providers is collected by Grid Resource Information Services (GRIS) and aggregated by a Grid Index Information Service (GIIS). Version 3 of MDS uses OGSi mechanisms, while version 4

uses mechanisms based on Web Services. The data components include GridFTP, an enhanced file transfer system supporting third-party transfers, striping and additional security features. The Open Grid Services Architecture Data Access and Integration project (OGSA-DAI), developed in collaboration with Globus, provides access to various database resources on the Grid [99]. The main computing resource component is the Globus Resource Allocation Manager (GRAM). The GRAM Gatekeeper receives requests written in a Resource Specification Language. If the request is authorised and can be fulfilled it is passed to a job manager, which passes the work to a batch queueing system. Early versions of the Globus Toolkit, up to and including version 2, used communications protocols which were well-documented but otherwise non-standard. The Globus developers should not be faulted for this: at the time practically every class of networked software had its own protocol. From version 3 onwards there was a clear shift to developing services as Web Services which operate over HTTP or HTTPS on the standard ports of 80 and 443. Web services bring certain benefits such as implementation independent and machine readable (and semi-humanreadable) protocol specifications in the form of WSDL and circumvention of port-based firewall restrictions. Version 3 required a modified Web Services engine to host the Globus Grid services. Version 4 aimed to produce standards compliant Web Services that could be used in any compliant Web Services container.

Security in GT Globus uses X.509 certificates for authentication and can support a variety of authorization systems. The simplest authorization system uses a Grid map-file to map certificate subject names to local account names. Another system is the Community Authorization Service. A string representation of the certificate subject name is used to perform this mapping. The string representation can cause problems with names that use unusual distinguished name components and the support for non-

ASCII characters is rather poor. The certificate's issuer name is not included in the mapping; it is assumed that different CAs will not issue certs with the same subject distinguished name to different entities and this uniqueness requirement is enforced by the agreements on CA namespaces.

In the context of Globus each virtual organization is expected to set up its own Grid infrastructure. The Globus software is not explicitly aware of VOs since it is assumed that all communicating users and services are within the same VO.

Single Sign-on Delegation: A goal of the Globus Grid Security Infrastructure (GSI) was to provide a 'single sign-on' capability for users as shown in Figure 2.4 [4]. This means that a user must sign on to the Grid only once per session no matter which resources are used. Another requirement was to allow delegation of authority from a user to a Grid service acting on the user's behalf. This takes the form of delegation of the user's identity. In Globus GSI, single sign-on and delegation are accomplished by the use of proxy credentials. To implement single sign-on, a new key-pair is generated on the user's system and a certificate is created with identity details based on the user's own certificate. The proxy certificate is signed with the user's private key. The proxy credential's private key is stored unencrypted and so the user does not need to re-enter the pass-phrase of their long-term private key. To delegate a credential to a remote service the user and the service mutually authenticate and the user indicates a wish to delegate. The service generates a new key-pair and returns a certificate signing request for the user to sign. The user signs the request with their private key to create a proxy certificate and returns it to the service. The private key of the delegated proxy credential is not transferred across the network. Proxy credentials are a form of impersonation and so effectively have all the rights of the originating user. In addition, the delegated private keys are typically stored unencrypted and protected solely by

the operating system's own measures. Although this form of delegation allows a great amount of flexibility it also reduces the level of security by increasing the risk of theft of the delegated credentials. To limit the exposure, a proxy certificate has a short lifetime (typically 12 or 24 hours). Revocation of proxy certificates is not directly supported. The short lifetime can, however, cause problems for long-running Grid jobs and so in practice mechanisms for renewing proxies are required. The proxy certificate may contain a flag to indicate that further delegation is not permitted. In the validation of this proxy certificate, the basic Constraints attribute of the users certificate (which states that it is not a CA certificate) is deliberately ignored, going against normal validation procedures. Proxy certificates are widely used by other Grid software which is based on, or interoperates with, Globus.

Community Authorization Service: The Community Authorization Service (CAS) is responsible for managing the policies that govern access to the resources of a virtual organization or community. The service addresses the requirements for scalability in administering VOs; flexibility in the expression of requirements in policy and allows for hierarchical policy management. The service keeps a record of membership of a community as well as the relevant policies. A user wishing to access a resource contacts the CAS server and requests the appropriate rights. The server checks the policies and grants appropriate capabilities to the user, which can then be presented to the resource. The resource examines the capability and grants access if the request is authorised. Capabilities are issued in the form of certificate extensions contained in proxy certificates. The capability policy and an indication of the policy language used are included. Such a credential is known as a restricted proxy. The CAS service does not dictate any particular policy language.

With a detailed discussion on Globus Toolkit as most powerful and robust middleware,

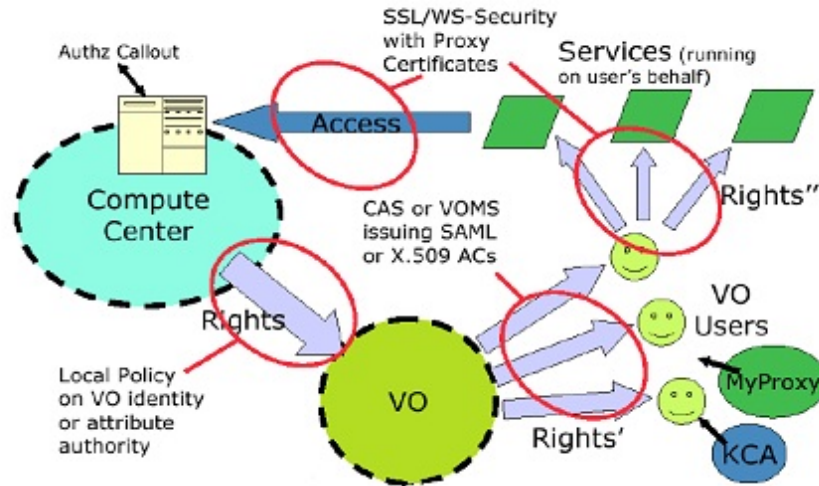


Figure 2.4: Grid Security Infrastructure [4]

the next subsection highlights the state of Security in UNICORE.

2.4.2 UNICORE

Uniform Interface to Computing Resources (UNICORE) is a project funded by German Ministry of Education and Research. It offers a ready-to-run Grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet. The latest version of UNICORE include features such as job execution environments, an enhanced generic application Graphical User Interface (GUI) and interactive site access via Secure Shell (SSH). The UNICORE project took advantage of the languages, protocols and standards used on the Web at the time of its initial development such as Java, HTML and HTTPS. In the UNICORE model, a piece of work is described in an Active Job Object (AJO).

Security in UNICORE: X.509 CAs provide certificates for UNICORE users and services as shown in Figure 2.5. These are used to authenticate users and to digitally

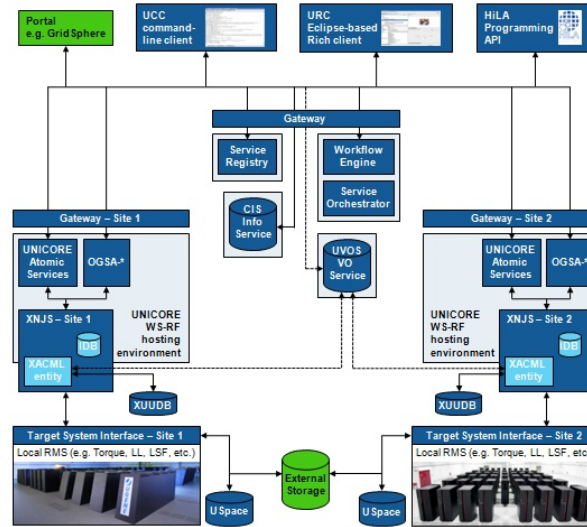


Figure 2.5: Unicare Security Architecture [222]

sign the UNICORE applets. AJOs are transferred using the UNICORE protocol layer (UPL), based on SSL. When an AJO is transferred to a server it must be accompanied by two certificates; one from an endorser and the other from a consignor. The endorser corresponds to the originating user and the AJO is signed by the endorser. The consignor is the entity transferring the job. When a user submits an AJO initially they will act as both endorser and consignor. A server may consign the AJO to another server but the server can only consign jobs that have already been endorsed. The destination server will verify the endorser's signature and make an authorization decision based on the identity and the local policy. Explicit Trust Delegation has been proposed to make delegation more flexible. A user role is introduced which identifies the originating user, but the endorser role can now be delegated to an agent or server. This allows the agent to create actions and AJOs on behalf of a user in a controlled manner.

2.4.3 Condor

Condor is a software system that creates a High-Throughput Computing (HTC) environment. It effectively utilizes the computing power of workstations that communicate over a network. Condor can manage a dedicated cluster of workstations. Its power comes from the ability to effectively harness non-dedicated and preexisting resources under distributed ownership.

Security in Condor Condor does not run user jobs in sandboxes. Condor assumes that users are trustworthy. Condor can prevent unauthorized access to the Condor pool, to help ensure that only trusted users have access to the pool. In addition, Condor provides encryption and integrity checking, to ensure that data (both Condor's data and user job's data) has not been examined or tampered with while in transit.

Condor's Security Model: At the heart of Condor's Security model is the notion that communications are subject to various Security checks. A request from one Condor daemon to another may require authentication to prevent subversion of the system. A request from a user of Condor may need to be denied due to the confidential nature of the request. The Security model handles these example situations and many more. Requests to Condor are categorized into groups of access levels, based on the type of operation requested. The user of a specific request must be authorized at the required access level. For example, executing the condor status command requires the READ access level. Actions that accomplish management tasks, such as shutting down or restarting of a daemon require an ADMINISTRATOR access level. There are two sides to any communication or command invocation in Condor. One side is identified as the client and the other side is identified as the daemon. The client is the party that initiates the command and the daemon is the party that processes the command and responds. In some cases, it is easy to distinguish the client from the daemon, while in

other cases it is not as easy. Condor tools such as `condor submit` and `condor config val` are clients. They send commands to daemons and act as clients in all their communications. For example, the `condor submit` command communicates with the `condor schedd`. Behind the scenes, Condor daemons also communicate with each other; in this case the daemon initiating the command plays the role of the client. For instance, the `condor negotiator` daemon acts as a client when contacting the `condor schedd` daemon to initiate matchmaking. Once a match has been found, the `condor schedd` daemon acts as a client and contacts the `condor startd` daemon. Commands in Condor are executed over TCP/IP network connections. While network communication enables Condor to manage resources that are distributed across an organization (or beyond), it also brings in Security challenges. Condor have ways to ensure that commands are being sent by trustworthy users. Jobs that are operating on sensitive data must be allowed to use encryption such that the data is not seen by outsiders. Jobs may need assurance that data has not been tampered with. These issues can be addressed with Condor's authentication, encryption and integrity features. The process by which Condor determines the Security settings that will be used when a connection is established is called Security negotiation. Security negotiation's primary purpose is to determine which of the features of authentication, encryption and integrity checking will be enabled for a connection. In addition, since Condor supports multiple technologies for authentication and encryption, security negotiation also determines which technology is chosen for the connection. Security negotiation is a completely separate process from matchmaking and should not be confused with any specific function of the `condor negotiator` daemon. Condor supports Password, SSL, GSI, Kerberos, File system, File system Remote, Windows and Anonymous authentication. The authorization portion of the Security of a Condor pool is based on a set of configuration macros. The macros

list which user will be authorized to issue what request, given a specific access level. When a daemon is to be authorized, its user name is the login under which the daemon is executed. These configuration macros define a set of users that will be allowed to (or denied from) carrying out various Condor commands. Each access level may have its own list of authorized users. Encryption provides privacy support between two communicating parties. Through configuration macros, both the client and the daemon can specify whether encryption is required for further communication.

2.4.4 SUN Grid Engine

The SUN Grid Engine is an advance resource management tool for heterogeneous and distributed computing environment. It orchestrates the delivery of computational power, based on enterprise resource policies that the administrator manages. The system uses these policies to examine available computer resources in the Grid. The system gathers these resources and then it allocates and delivers them automatically, in a way that optimizes usage across the Grid.

Security in SUN Grid Engine The SUN Grid engine system has the following four categories of users:

- (i) **Managers:** Managers have full capabilities to manipulate the SUN Grid engine system. By default, the superusers of the master host and of any machine that hosts a queue instance have manager privileges.
- (ii) **Operators:** Operators can perform many of the same commands as of managers, except that operators cannot add, delete or modify queues.
- (iii) **Owners:** Queue owners are restricted to suspending and resuming or disabling and enabling, the queues that they own. Users are commonly declared to be

owners of the queue instances that reside on their desktop workstations.

- (iv) Users: Users have certain access permissions, as described in configuring Users, but users have no cluster or queue management capabilities.

To enable cooperation in the Grid, the following should be considered by the administrator:

- Negotiate policies.
- Ensure that policies for manual overrides for unique project requirements are flexible.
- Automatically monitor and enforce policies.

The user names must be declared before defining the share-based, functional or override policies for users. The status can be seen while configuring Policy-Based Resource Management With QMON. If user names are not declared explicitly before defining policies, the Grid engine system can automatically create users, based on predefined default values. The automatic creation of users can significantly reduce the administrative burden for sites with many users. An administrator can define high-level usage policies that are customized for the site. Four such policies are available:

- (i) Urgency policy
- (ii) Share-based policy
- (iii) Functional policy
- (iv) Override policy

Policy management automatically controls the use of shared resources in the cluster. High-priority jobs are dispatched, preferentially. These jobs receive greater CPU entitlements when they are competing with other lower-priority jobs. The Grid engine software monitors the progress of all jobs. It adjusts their relative priorities correspondingly and with respect to the goals that are defined in the policies.

This policy-based resource allocation grants each user, team, department and all projects an allocated share of system resources. This allocation of resources extends over a specified period of time, such as a week, a month or a quarter.

2.4.5 LEGION

Legion is an object-oriented distributed computing system, which was developed at the University of Virginia. The Legion design aims to provide a secure, easy-to-use, high performance system for parallel processing which would take advantage of resource heterogeneity. From a user's perspective, the aim is to provide a single virtual computer with a consistent object name space and transparent access to distributed resources. Legion fully support existing codes written in Message Passing Interface (MPI) and Parallel Virtual Machine (PVM). Legion's object-based architecture dramatically simplifies building add-on tools for tasks such as visualization, application steering, load monitoring and job migration [223].

Security in Legion The Legion project made some clear security assumptions:

- (i) That the system will be running across a very wide area
- (ii) That it will be running on top of different host systems, which may get compromised or even be malicious.

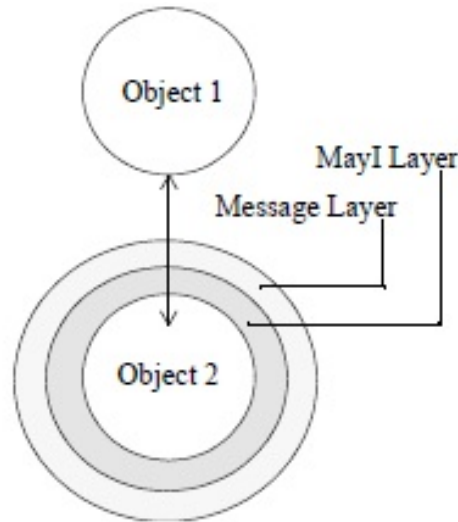


Figure 2.6: Legion Architecture [223]

- (iii) That there will not be a single owner of all the systems involved and so there may be varying policies and levels of security.

From these a number of high-level principles are developed. The first principle means that Legion should not significantly increase the risk to a resource. The second principle means that users are responsible for their own security policy. The third principle means that a simple security model is to be preferred. In the Legion model, every object has a MayI method which determines if a particular call is permitted as shown in Figure 2.6. There is a distinct set of rights attached to each of the object's methods. The 'I' in question can be the Calling Agent, who initiated the call in question; the Responsible Agent, often the ultimate 'user' who initiated the chain of events leading up to the call; or the Security Agent, a third-party policy enforcement agent. The MayI method can make a decision based on any subset of the Agents. The identity of a calling agent can be checked with an Iam method. Each object also has a Delegate method which is used to delegate particular rights to another object. Restrictions can be applied to the delegation. Every Legion object and user has a Legion Object Identi-

fier (LOID) which incorporates an X.509 certificate. This allows other objects or users to verify messages from or send encrypted messages to, an object or user with a known LOID. The default implementation of MayI checks the callers credentials against a list of LOIDs to accept or deny.

As, Legion is a distributed system, some familiar concepts of traditional monolithic system Security are different. For example, there is no central password file: each user's password is stored in the corresponding AuthenticationObject. Furthermore, any user can create an AuthenticationObject and create objects that no one else can call. For example, the system administrator may be the owner of an object that provides a resource such as printing. If that object looks in a particular group to determine access and the system administrator has not added a user to that group, access will be denied. Table 2.5 describes the comparison between above discussed middleware in terms of their support in the areas of authentication, authorization, interoperability and trust management. It shows that all the middleware have different authentication and authorization adopted, instead of a common solution, increasing complexities and interoperability needs. Interoperability is handled using different Interface Description Language, GRIP, Nexus communication library etc. Again, there is no standard solution available to handle interoperability. Trust management model is usually policy based which usually deals with identity trust. It poses the need to handle interoperability and trust management in Grid environments. Some popular Grid Projects are described here for their support for security:

European DataGrid: The European DataGrid Project [224] was set up to develop middleware for a data Grid to support scientific research. Data management, including handling of data replicated between sites, was one of the main goals of EDG. The Replica Location Service (RLS), developed in collaboration with Globus, provides

Table 2.6: Security in Grid Middleware

Middleware/ Features	Globus Toolkit	Unicore	Condor	Sun Grid Engine	Legion
Authentication	GSI Authentication using X.509 certificates and Mutual Authentication using SSL/TLS	Full X.509 certificates are used as base line	Condor provides support to authentication, integrity assurance through the use of configuration macros	Four types of users created for allowing access to resources	Defined authentication model. Uses AuthenticationObject to create certificate
Authorization	It allows for a variety of authorization schemes, including a Grid-mapfile access control list, an access control list defined by a service, a custom authorization handler, and access to an authorization service via the SAML protocol	Authorization decision is based on the identity of the user and the local policy	Requests to Condor are categorized into groups of access levels, based on the type of operation requested.	The SUN Grid Utility is designed with Solaris and network Security isolation for user jobs and with role based access principles. Incorporates the principal of minimum privilege for users and administrators	The Mayl Layer in Legion is responsible for access control
Interoperability	Uses Nexus communication library	GRIP is used for communicating with Globus	Interoperability is also job dependent. The job should use algorithms that can generate a large number of independent tasks that are self contained making them portable	Interoperability on all major Unix Platforms: Solaris, Mac OSX-HP-UX, AIX. Irix, Linux	Uses Interface Description Language (IDL) i.e. Compatible with multiple programming language, variation of RMI for communicating through LOIDs
Trust Management	Policy based	Policy based	Condor assumes that users are trustworthy and trust management is Policy Based	Policy based	Policy based
Scalability	Closer to Hardware and is more scalable	Limited Scalability through centralized broker, brokers hierarchy	Limited scalability as dedicated machines within an organization need to be used	SUN N1 Grid Engine can scale upto 10,000 hosts	Uses LOAs to form Computational hierarchy

mappings from logical names to replicated data. At the start of the project security requirements were collected and documented. These included seventeen requirements for authentication. Out of these, three important instances were: that a user should need to authenticate just once per session; that authentication should be interoperable between many Grids and applications; and that it should be possible to revoke authentication credentials in the event of loss or compromise. It was considered important to keep authentication and authorization separate. The primary reasons for this were to allow interoperability of authentication tokens with developing authorization systems and to allow single sign-on across multiple VOs. The authentication and delegation system is based on the Globus Grid Security Infrastructure, described above. An improvement within EDG was to create a virtual organization membership LDAP server which allowed VO information to be stored centrally. The EDG Local Centre Authorization Service (LCAS) is an authorization decision engine that can add advanced access control to underlying Grid services such as the Globus gatekeeper or GridFTP server. The EDG Local Credential Mapping Services (LCMAPS) provides a flexible system to map Grid users to local credentials on Grid sites.

CrossGrid: CrossGrid [225] was a sister project of EDG with an emphasis on interactive jobs: keeping the user in the loop. For example, two CrossGrid applications were emergency flood modelling and blood flow modelling for bypass surgery. CrossGrid enhanced the EDG resource broker with better match-making and support for advanced reservation, important features for allowing a user to interact with a submitted job in near real-time. A number of tracing, debugging and analysis tools were developed to aid Grid application writers and users. The EDG security policies and mechanisms were carried over to CrossGrid due to the shared software and objectives of the projects.

Large Hadron Collider Computing Grid: The Large Hadron Collider (LHC) at CERN is used for a number of large-scale particle physics experiments. The LHC Computing Grid (LCG) processes the output of these experiments [226]. The first version of the software for LCG was provided by EDG. Some of the more experimental (or unfinished) pieces of EDG were dropped for the sake of overall reliability. The LCG security architecture is based on that of the earlier European DataGrid software.

Enabling Grids for E-science: From its inception, EGEE [32] was seen as the follow-on project from EDG and a parallel project to LCG. Its primary goal was to run a production Grid service to support a number of scientific computing domains. A lightweight, portable Grid middleware, gLite, was to be developed, at least partly to reduce dependence on the Globus software. In 2006 the software releases from LCG and EGEE were merged. gLite 3.0 consisted of the latest version of the LCG software with a number of gLite components added. At the time of writing the LCG/gLite software still depends on external Grid and cluster components such as the Globus Toolkit, Condor-G and the Torque batch scheduler. The EGEE security architecture reflects the evolution of the earlier EDG architecture to take note of developments and in particular the move to Web Services and service-oriented architectures. The architecture aims to be modular, implementation agnostic and standards compliant. While web-services protocols can make use of message-level XML-based security, transport level is favoured because of the superior performance offered at the current state of development.

Grid Computing landscape is transforming and continuously evolving, posing complex Security requirements, issues and challenges at research community, to provide solutions for them. Most of the Security solutions available do not provide comprehensive models to address different types of Security requirements and access control policies.

Many of them are not based on open standards. Some of them are problem specific or use proprietary mechanisms. As observed during exhaustive review carried out it is clear that, no solution for Grid services exist that provides an answer to interoperability and trust management, as a whole. Most of the existing solutions address some of the parameters as observed in Table 2.5 and 2.6.

2.5 Problem Formulation

Based on the literature survey, it is apparent that as Grid is evolving there are various Security challenges which needs to be addressed before wide scale adoption of Grid. The resources in a Grid belongs to a multiple administrative domains, which are governed by their own Security standards and policies. Security functions required for such scenario are also quite dynamic. The Security of data and application assume critical importance as various industries and academia are involved in providing their resources for implementing cross-organizational Grid computing infrastructure. The existing Security solutions span across local domains to provide Security in them. However, in case of Grid, for services to traverse multiple domains there is a need for these multiple hosting environments to interact with each other, thus introducing the need for interoperability and enforcing trust at multiple levels. Grid systems involve the risk of executing transactions without prior experience and knowledge about each other's reputation and trust information of Grid participants. Recognizing the importance of trust in such environments, there is a necessity of designing strategies and mechanisms to establish trust in Grid environment. Based on this, the major issues identified for accessing resources in multiple domains are lack of interoperability and

trust management solutions within Grid systems.

This thesis makes an attempt to propose and implement Interoperability and Trust management Solutions for establishing secure Grids.

2.6 Objectives of the Thesis

The first objective is to study and explore key Security challenges in various Grid Computing Environments (GCEs). The second objective is to analyze interoperability and trust management issues like authentication and authorization in these GCEs. Third objective is to propose, design and implement interoperability and trust management solutions for GCEs. Fourth and the last objective is to test and validate the proposed interoperability and trust management solutions in GCEs.

To achieve the first objective, a comprehensive study of Grid Security requirements, challenges and issues has been done. A thorough study of standards and specifications used in Grid based systems has also been carried out.

To achieve the second objective, a thorough study and analysis of Interoperability and Trust Management issues such as authentication and authorization has been carried out. Work done in the area of authentication, authorization, interoperability and trust management in existing popular Grid middleware has also been studied in detail and reported as Literature Survey.

To achieve the third objective, four solutions, three interoperability: Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE) and one trust management: TUX-TMS (Thapar University Extensible Trust management System) have been proposed for addressing Interoperability

and Trust Management challenges respectively, in a Grid.

TUX-INTERO is a Web Services based interoperability solution which discovers and integrates resources based on various platforms and middleware. TUX-INTEROP is a grid based portal for secure interoperation of grids. PIE is grid portal which integrates resources from various middleware and allows access to end users at a single interface. These solutions fulfill the original vision of Grid Computing aimed at providing a single massive infrastructure to users instead of disparate resource islands.

TUX-TMS is a flexible Reputation based Trust Management system which combines the merit of policy proof and reputation evaluation. TUX-TMS establishes and evaluates trust among untrusted and unknown partners in a Grid. TUX-TMS is a transaction based system where a user is not allowed to access service again unless response in the form of feedback is given. Various parameters were considered for establishing and evaluating trustworthiness and reputation of domains such as Trust Context Factor, Reputation Decay, Trustworthiness Decay, Risk Assessment and Trust Inheritance. Trust relationships of direct, indirect and recommended are considered to further evaluate the trustworthiness of target service/resource in case of ambiguity. It also presents a Identity Management System to store, retrieve and update multiple user credentials. This system provides incentive to resource/service providers to contribute resources as well as motivates resource/service users to interact and transact depending on their reputation and trustworthiness information. Binding trust information and providing a threshold for accessing Grid services encourages minimal chances of malicious use or security attacks on resources. The taxonomy, architecture and implementation details of TUX-INTERO and TUX-TMS have been presented in this thesis as chapter 3 and chapter 4.

To achieve the fourth objective, TUX-CITMS, TUX-INTEROP and PIE has been de-

veloped as a Grid Testbed at Thapar University. TUX-INTERO and TUX-TMS have been integrated, deployed and tested on TUX-CITMS taking various test cases. The experimental details and testing results have been reported in Chapter 5. The complete proposed Interoperability and Trust Management solutions have been evaluated by implementing different security related scenarios. These scenarios and implementations cover different aspects related to authentication, authorization, interoperability and trust management. The results show that the various implementations are able to meet the identified security challenges. The results clearly demonstrate that the proposed solutions are workable and can be effectively used to address key security challenges related to interoperability and trust management to establish secure Grid environments.

To conclude, results of TUX-CITMS, TUX-INTEROP and PIE show that the proposed Interoperability and Trust Management solutions can achieve high transaction success rate by providing secure services. The thesis clearly depicts that a collaborative interoperability and trust management solution enhances grid usage securely.

In summary, the sharing that the Grid environment is concerned with, is not primarily file exchange, but rather direct access to computers, software, data and even other kind of resources. The heterogeneity and dynamicity of the resources involved in the Grid environment requires considerable efforts for securing them. This chapter focused on exploring key Security challenges faced by Grid Computing environment. It analyzed interoperability and trust management issues such as authentication and authorization in various Grid Computing environments.

Chapter 3

The Proposed Interoperability Solutions: TUX-INTERO, TUX-INTEROP and PIE

The previous chapter discussed the research work accomplished in the area of Grid Security Challenges. The Literature Review depicts that Interoperability and Trust Management challenges have not been fully addressed. To address Interoperability challenges in grids, Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE) have been proposed, designed, implemented and tested. TUX-INTERO is a Web Services based interoperability solution which discovers and integrates resources based on various platforms and middleware. TUX-INTEROP is a grid portal for secure interoperation of grids. PIE is grid portal which integrates resources from various middleware and allows access to end users at a single interface. This chapter discusses taxonomy, architecture and implementation details of the proposed Interoperability Solutions.

3.1 Interoperability

Today, Grid Computing exists in the form of disparate resource islands and infrastructures, in contrary, to how Grid Computing was envisioned. When it comes to unifying these massive Grid infrastructures the foremost challenge to be addressed is interoperability. Interoperability is the ability of Grid-systems to exchange information and use the information that was exchanged. Existing approaches available to address interoperability are: use case scenario, parallel deployment, gateways, adapters and translators, as discussed in Chapter 1. These approaches can be considered while designing an Interoperability system or solution. Web services are emerging as a promising paradigm for heterogeneous and distributed computing systems.

Taking into consideration the approaches available to address Interoperability in grids, TUX-INTERO, TUX-INTEROP and PIE have been proposed as Interoperability solutions in this research work. TUX-INTERO is an endeavor to provide Grid users not only the freedom of higher operability to access resources but also security within a Grid environment, leading to realization of original vision of Grid as a single massive infrastructure. TUX-INTEROP and PIE have been proposed as grid portals based on different set of portlets for pluggability and extensibility. The next section discusses TUX-INTERO in detail.

3.2 TUX-INTERO: A Web Services based interoperability solution

TUX-INTERO is a Web Services based interoperability solution whose primary function is to allow users access to heterogeneous resources hosted on multiple platforms. TUX-INTERO facilitates resource information publication, management and brows-

ing. The knowledge of resources is essential to any Grid system, making information services, an important component. The ability to discover resources available for a particular task requires that resource capabilities are published and are retrievable by Service Requesters. In addition to capabilities, resources must also publish contact information such that they can be invoked. Finally, it must be possible to discover the dynamic state of a resource to determine the optimal resource at any time. Before discussing TUX-INTERO in detail, some terms concerning TUX-INTERO are discussed as taxonomy in next section.

3.2.1 Taxonomy of TUX-INTERO

Following taxonomy has been used in context of TUX-INTERO:

- (i) Web service: A Web Service is a self contained, self describing modular component that can be published, located and invoked across the web.
- (ii) Service: A Service can be defined as a computation that may be performed in response to a request.
- (iii) Service Description: The mechanisms required to describe the entry points of a particular service along with specification of messages/documents it is able to process. Then entry points are usually described using interfaces. These are similar to interfaces known from object oriented programming languages. Informally, interfaces may be considered to be sets of message types which together model some coherent functionality. By implementing an interface, the service guarantees that it understands the message types defined by this particular interface.
- (iv) Service Discovery: The process in which a required service is located. The user who is locating a service for later usage is usually referred to as the Service

Requester. The criteria a Service Requester may specify to locate a service does not depend on a particular technology/implementation and is not predefined.

- (v) Service Integration: The process of communication between a Service Requester and a service is termed integration since the service is usually integrated into one or more complex applications (which may consist of more services).
- (vi) Service Provider: The owner of the service is known as Service Provider. From an architecture perspective, this is the platform that hosts access to the service. The Service Provider is built using Simple Object Access Protocol (SOAP). The main benefit from using SOAP is multi-language supporting. Since there is support for the SOAP protocol in many programming language, for example, C++, Java and VB, application developers have flexibility of selecting the programming environment of their choice. In addition to the standard SOAP client infrastructure, the system clients need XML message building and parsing capability depending on the query and response specification.
- (vii) Service Requester: The consumer of the service is known as Service Requester. From an architecture perspective, this is the application that is invoking or initiating an interaction with a service. A program or other Web Services can play this role.
- (viii) Publish: In order for a service to be accessible, a service description needs to be published so that the Service Requester can find it. The published information can vary as per service description.
- (ix) Find: The Service Requester retrieves a service description from the service registry in order to find the required resource. The find operation can be involved

in two different phases for Service Requester: at design time to retrieve the service's interface description for program development and at runtime to retrieve the service's binding and location description for invocation.

- (x) Bind: The service needs to be invoked. In the bind operation the Service Requester invokes or initiates an interaction with the service at runtime using the binding details in the service description.

In TUX-INTERO, service descriptions play a central role in maintaining a loose coupling between Service Requesters and Service Providers. The service description, published by the Service Provider, allows Service Requesters to bind to the Service Provider. The Service Requester obtains service descriptions through a variety of techniques, from the simple “e-mail me the service description” approach to various other approaches available.

This section discussed the taxonomy of TUX-INTERO paving a way to discussing TUX-INTERO architecture.

3.2.2 Architecture of TUX-INTERO

TUX-INTERO utilizes a storage component- a database with gateways connected to the underlying native information services to retrieve resource information and publish to users. It scales with the number of supported information providers, while being independent of any privileged middleware. There is a large set of Service Providers. Each of them uses a schema described with a logical formalism to store knowledge about the local resources and background knowledge about resources in general. This background knowledge contains links between different schemas, enabling mediation between these schemas. The overall amount of information stored in the network is

large. Both the set of Service Providers and the knowledge each Service Provider stores is highly dynamic. A user can formulate a query using one or multiple schemas from some of the Service Providers and receive a list of matching resources from all Service Providers in response, incorporating also knowledge which can be derived by using background knowledge and mediating knowledge from all Service Providers. If the user specifies sorting criteria, the result set is a top k list.

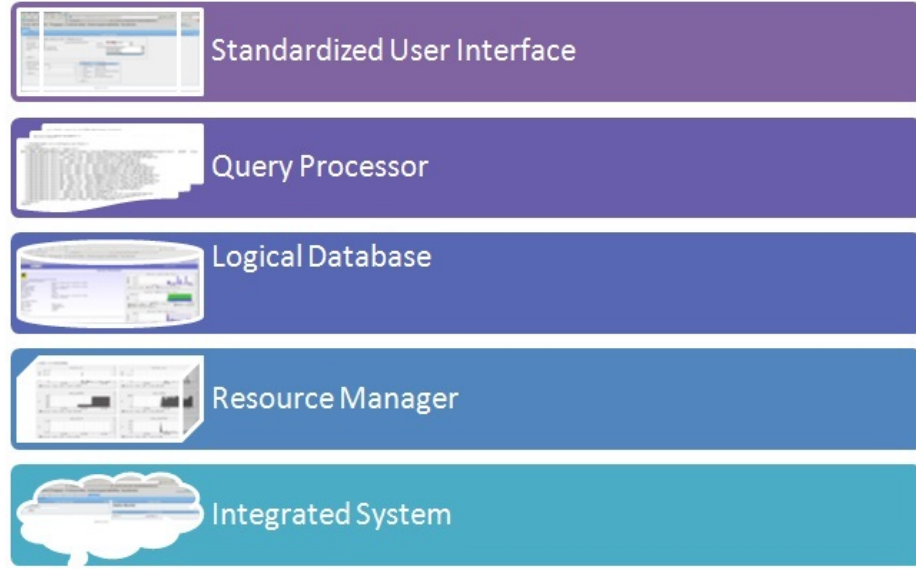


Figure 3.1: High Level Architecture of TUX-INTERO

The high level architecture of TUX-INTERO is presented in Figure 3.1 and detailed in Figure 3.2. The five layered architecture is composed of various components which are discussed here, starting from lowest layers to the top.

3.2.2.1 Integrated System

Integrated System lies at the lowest layer in TUX-INTERO architecture. The Integrated System is composed of clusters running on various platforms. The Integrated system uses Web Services to integrate nodes on various platforms. It simply integrates the resources available and provide the details in pair of {grid, cluster, node} to the

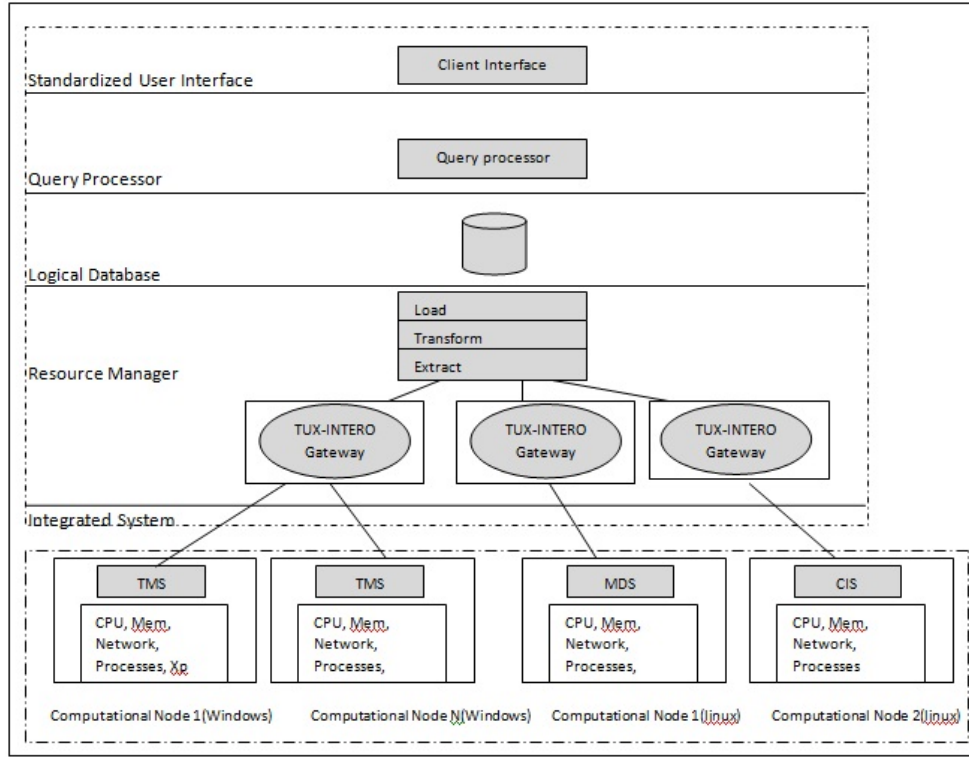


Figure 3.2: Detailed Architecture of TUX-INTERO

the Resource Manager, where grid is the name of Grid testbed, cluster is the name of Cluster's network address and node is the agent representing the information provider node. TUX-INTERO considers grid as same for all as it is located at Thapar University Campus Grid, cluster and nodes are dependent as per availability of the resources belonging to various domains. Figure 3.3 shows Class Diagram of Grid entities. Integrated layer depicts the physical resources available. The second layer is comprised of Resource Manager component.

3.2.2.2 Resource Manager

The Resource Manager is an interface between middleware-specific information services and the TUX-INTERO information database. It involves a five step process of extracting, parsing, checking, transforming and loading (as shown in Figure 3.4). The first

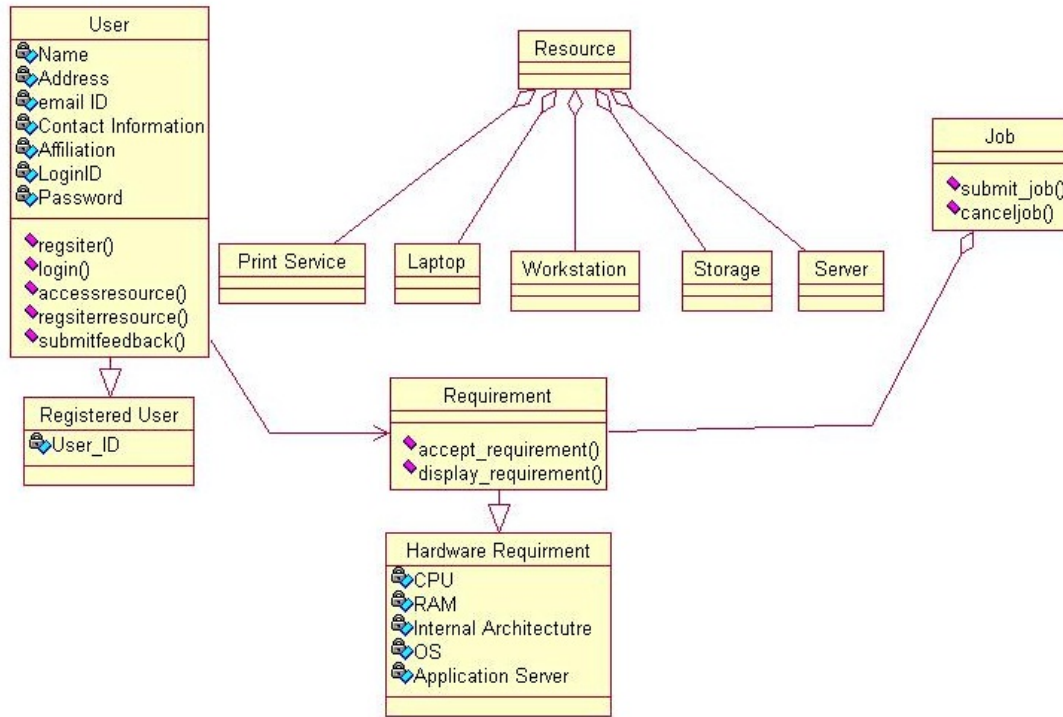


Figure 3.3: Class Diagram of Grid entities in TUX-INTERO

step extracts resource information from Grid middleware specific extractors and then extracted Grid resource information is parsed, checked and transformed in a flexible way into a common data structure, for example, the GLUE 2.0 schema. This process of extracting data from information services, transforming this data to the GLUE 2.0 schema and loading it into the target database is realized by Web Services. The third layer is comprised of Logical Database component.

3.2.2.3 Logical Database

The data usually consists of information about resources and services that exist within the Grid infrastructure. Converting this mapping into syntactically and semantically fitting data structures and integrating it with the homogeneous data model lays the basis for monitoring of resources and services. The identifier of resources and services

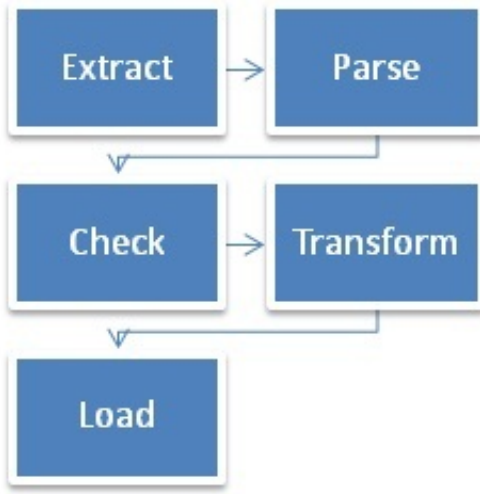


Figure 3.4: Resource Manager in TUX-INTERO

recorded in the resource management system as well as in the resource monitoring services must be unique and identical in order to allow a deterministic mapping of data sets to views. The database is taken here as MS-SQL. The third layer is comprised of Query Processor component.

3.2.2.4 Query Processor

The larger the Grid grows, the more resources are described. Therefore, there might be huge result sets for queries which are rather generic. This makes it infeasible to deliver all results of a query to the user. In these circumstances, a user may not be typically interested in every result. Users might just seek for best resource matching their requirements. In this case, simply delivering the first result which have been found is enough. In other cases, the user has preferences according to some attributes of the resource. Thus, there needs to be a way for the user to express a desired ranking of the result set. Typical criteria might be the price for resource usage, the network distance from the user, either in terms of latency or bandwidth, or the geographic distance.

Other criteria include quality of service aspects as security guarantees or performance guarantees. Resource descriptions in the Grids grow on the one hand because more resources and types of resources are integrated and on the other hand because the increasing heterogeneity demands for a very detailed resource description. Thus, the system architecture and the query strategies must cope with these amounts of data. For each query, the TUX-INTERO needs to know the identity of the querying entity (user or service) that wants to retrieve the data. Newer methods use a combination of Virtual Organization Membership Service (VOMS) and Security Assertion Markup Language (SAML) to get the credentials of an accessing entity. As soon as credentials of a querying entity are known, the query can be filtered and the corresponding data sets can be provided. The fourth layer is comprised of Standardized User Interface component.

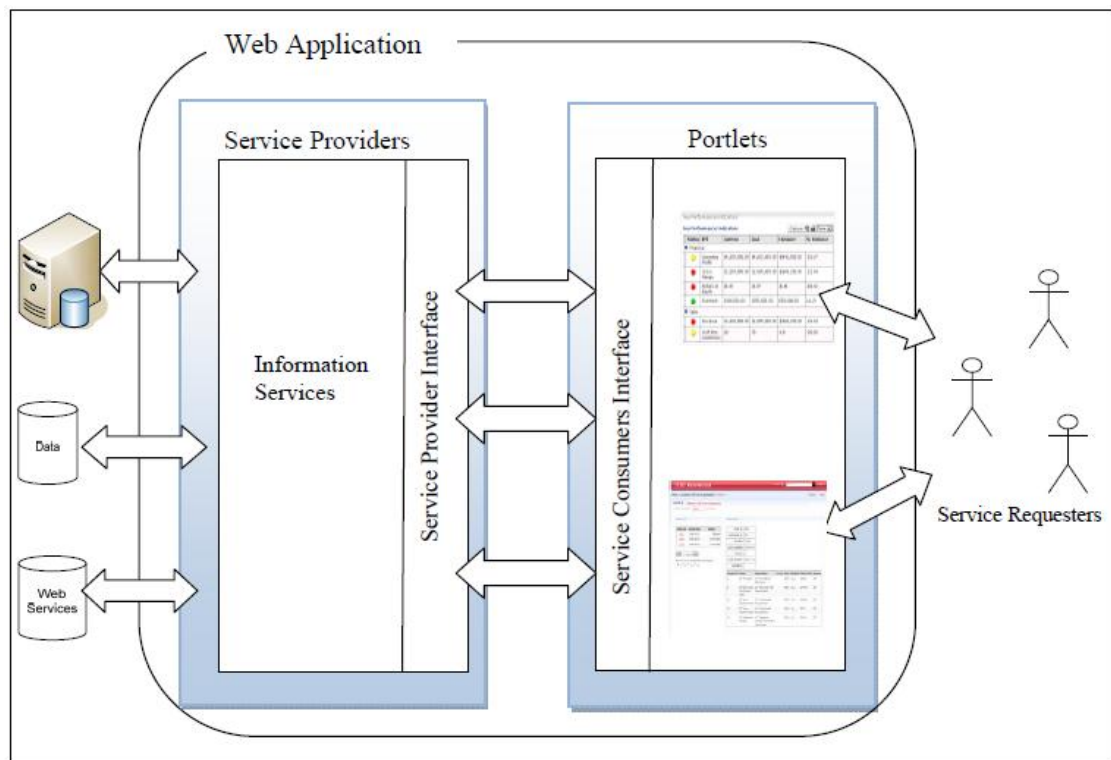


Figure 3.5: High level Architecture of TUX-INTEROP User Interface

3.2.2.5 Standardized User Interface

A standardized user interface augments resource information by querying the database for resources available for a particular application as shown in Figure 3.5. According to the Grid requirements, the standardized database access interface must be compliant with the Open Grid Services Architecture Data Access and Integration (OGSA-DAI) specification. An overall architecture should account for scalability and efficiency, which includes minimal resource requirements and short access time to current information. The core architecture of TUX-INTERO is an autonomous and middleware-independent service which utilizes a storage component with gateways connected to the underlying native information services.

After discussing TUX-INTERO architecture into detail, the next section elaborates implementation details of TUX-INTERO.

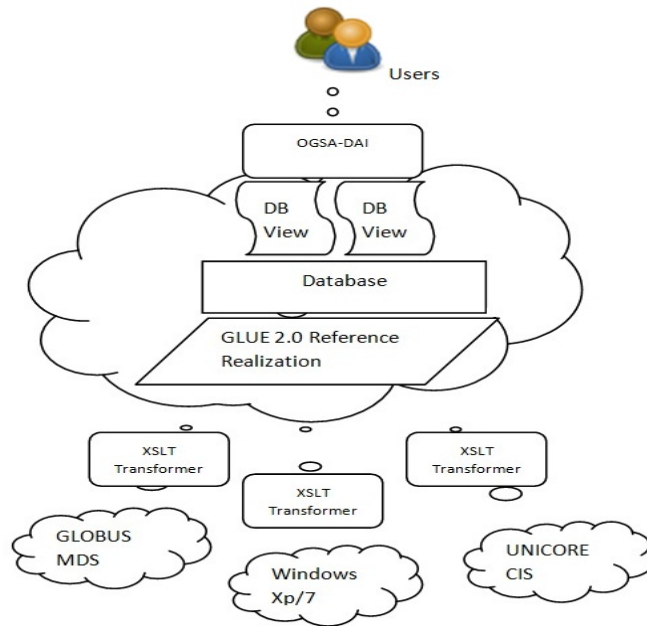


Figure 3.6: Deployment of TUX-INTERO

3.2.3 Implementation details of TUX-INTERO

TUX-INTERO aims at collaborating heterogeneous resources and making use of them (as shown in Figure 3.6). The architecture unveiled the functioning and components at various layers in TUX-INTERO. This section discusses the implementation details of TUX-INTERO are presented below.

3.2.3.1 Collaborating resources from varied domains

The bottom layer represents clusters of resources on various middleware such as Globus, Unicore, etc. Firstly these resources are integrated using ASP.Net Web Services to create a Thapar University Campus Grid. In Thapar University Campus Grid, Service Providers can easily publish resources by invoking Web services to pass information to the Application Server. Clients are developed independently for various platforms to invoke *ProviderService* Web Service. The *TMSService* is the web reference added to invoke *ProviderService* Web Service for Windows platform. *ProviderService* is hosted in Windows platform and designed using ASP. Net tools. *ProviderService* is responsible for updating node information to the Application Server on the availability of resources using two web methods *ServiceOnline()* and *ServiceOffline()*, explained below. TUX-INTERO further demonstrates interoperability with Globus and Unicore clients by using information from respective Information Services. The *ProviderService* Web Service is developed in ASP.Net and invoked using clients developed for Windows and Linux systems independently. The Integrated system is composed of nodes running on Windows and Linux platforms. The Integrated system dynamically collects information from multiple resource hosted on multiple platforms in two steps. In the first step all the information is gathered using SIGAR API's classes. SIGAR (System Information Gatherer and Reporter) is a cross-platform, cross-language library

and command-line tool for accessing operating system and hardware level information in Java, Perl and .NET. Then, after each client passes information to the Application server using a SystemInfo Web Service, the data is updated. The implementation details are elaborated below:

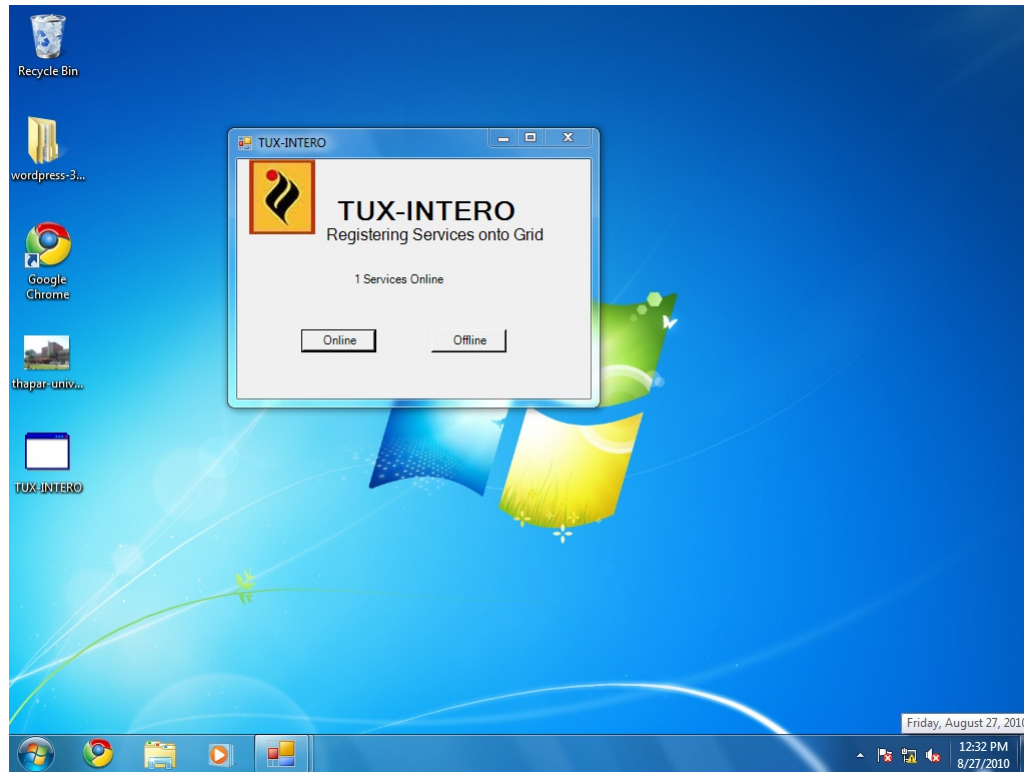


Figure 3.7: Desktop Application designed in .Net for invoking ASP.Net Web Services in TUX-INTERO

- (a) For a Windows based node: A desktop application TMS (Figure 3.7) is developed in ASP. Net which invokes *ProviderService* Web Service (Figure 3.8). The *ProviderService* updates the information to Application Server for the availability of the resources. Here, the web methods `ServiceOnline()` and `ServiceOffline()` are used to pass the resource information, as mentioned below:

At the Service Provider End: A desktop application is designed and implemented in ASP.Net, which supports and invokes these basic SOAP invocation

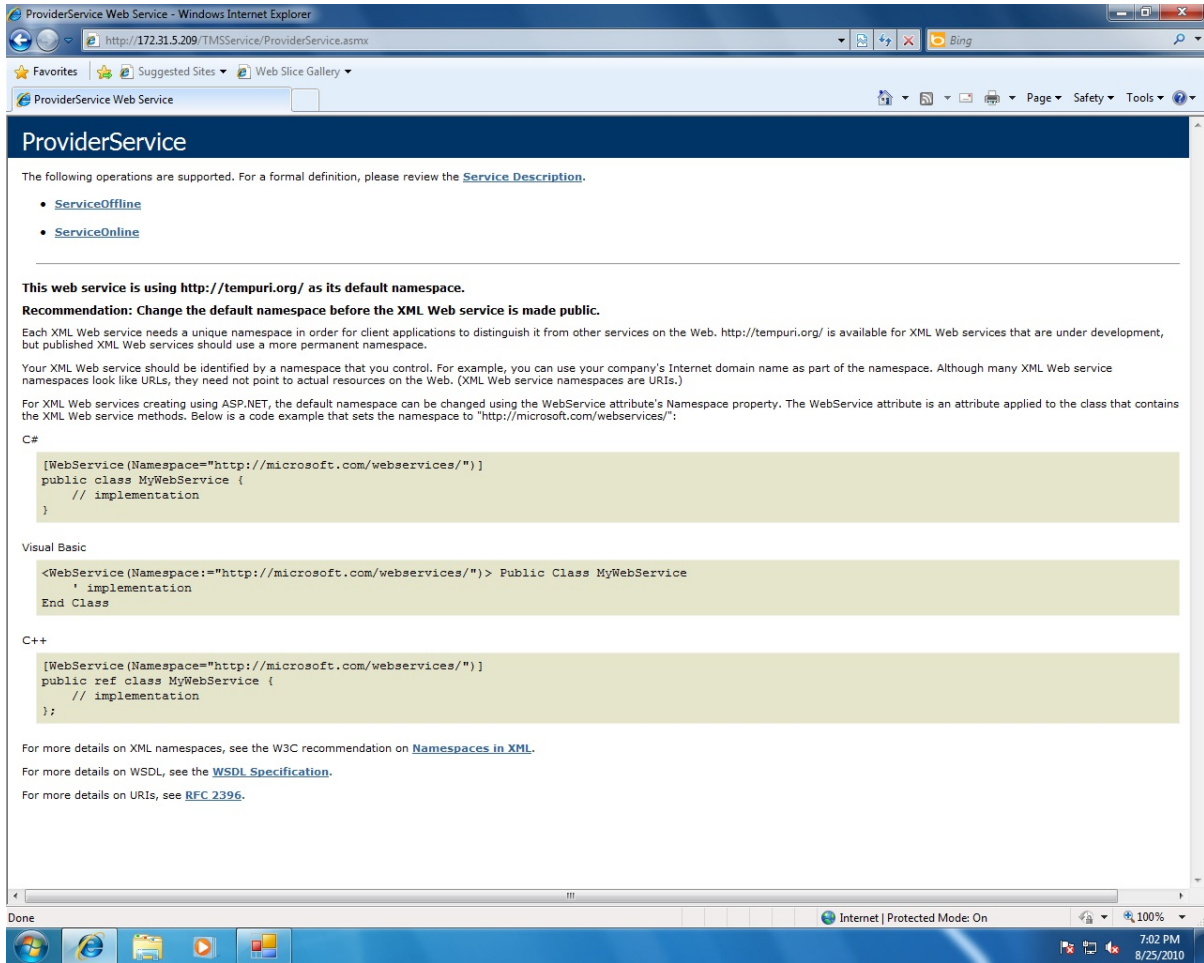


Figure 3.8: Invoking Web Services in TUX-INTERO

methods. The web methods for logging information are described here.

- **ServiceOnline():** Publishes system and providers information and updates the information in the database which is further updated in Application Server. It shows the Provider is available to volunteer services and become part of the Grid. Web Services Description Language (WSDL) is generated when ServiceOnline() web method is invoked. *Provider_Id* and ID are passed as parameters to the Web Service.
- **ServiceOffline():** This web method deletes system information to indicate Service Provider has been shutdown or is not willing to share its information.

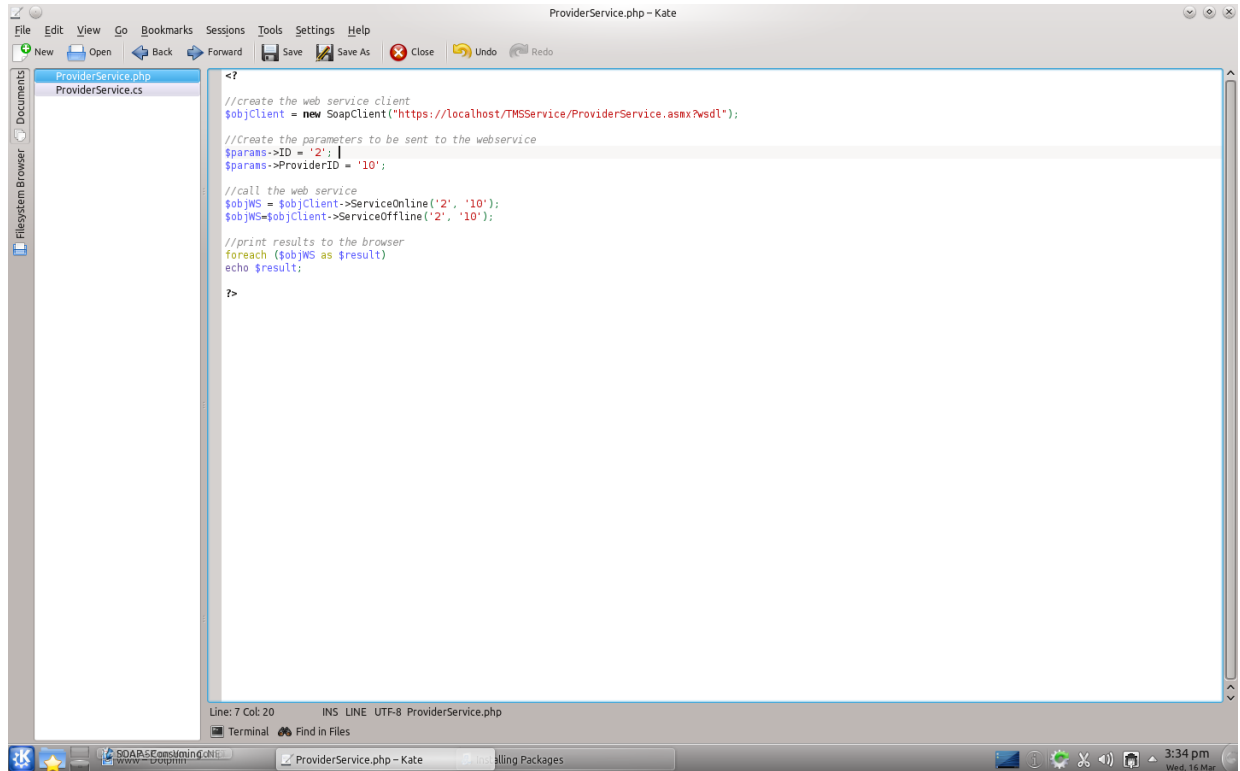


Figure 3.9: Invoking ASP. NET Web Services Using PHP

System name is provided as parameter to this web method. WSDL is generated when ServiceOffline() web method is called is shown in Figure 3.8.

At the Service Consumer End: Authenticated Consumers can retrieve list of resources available on system connected to Grid from status database by specifying their requirements.

- (b) For a Linux based node: An application *TMS_Linux* is developed in PHP and implemented which adds reference to invoke ProviderService.asmx Web Service as shown in Figure 3.9. Here, the web methods are ServiceOnline() and ServiceOffline() are used to pass resource information, as mentioned below:

At the Service Provider End: An application is designed and implemented in PHP, which supports and invokes these basic SOAP invocation methods. The web

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:dsml="http://www.dsml.org/DSML">
<!--Using the W3C XSLT and DSML namespaces -->

<!--Creating the text based output -->
<xsl:variable name="informationProvider">CIS</xsl:variable>
<xsl:variable name="sourceAddress">thapargrid-thapar.grid.edu</xsl:variable>
<xsl:variable name="sourceAddress">thapargrid-thapar.grid.edu</xsl:variable>
<xsl:variable name="insertTime">CURRENT_TIMESTAMP</xsl:variable>

<!--Checking for valid ids -->
<xsl:template match="/">
<xsl:apply-templates select="//thapar:thapar/thapar:directory-entries/thapar:entry/">
</xsl:template>

<!--A template for Thapar entries -->
<xsl:template match="//thapar:thapar/thapar:directory-entries/thapar:entry">
<xsl:variable name="DistinguishedName"><xsl:value-of select="@dn"/></xsl:variable>
<xsl:choose>

<!--GlueSiteUniqueID -->
<xsl:when test="starts-with(@dn,'GlueSiteUniqueID')">

<!--Parsing the Unique ID of the Admin Domain-->
<xsl:variable name="DistinguishedNameLocal">
<xsl:value-of select="substring-after(substring-before(
    $DistinguishedName,'mds-v0-name=local,o=grid'),'o-name=')"/>
</xsl:variable>

<!--Creating SQLoutput for parsing -->
INSERT INTO GLUE20.AdminDomain (
ID,Name,Description, Distributed, informationProvider,sourceAddr,insertTime,trustworthiness,reputation)
VALUES ("<xsl:value-of select="$DistinguishedNameLocal"/>","
"<xsl:value-of select="dsml:attr[@name='GlueSiteName']/thapar:value"/>","
"<xsl:value-of select="dsml:attr[@name='GlueSiteDescription']/thapar:value"/>","
"<xsl:value-of select="thapar:attr[@name='GlueSiteeotherInfo']/thapar:value"/>","
"0",
<xsl:value-of select="$informationProvider"/>","
<xsl:value-of select="$sourceAddress"/>","
<xsl:value-of select="$insertTime"/>)
ON DUPLICATE KEY
Name=VALUES (Name)
Description=VALUES(Description),
Distributed=VALUES(Distributed),
sourceAddr=VALUES(sourceAddr),
insertTime=VALUES(insertTime);

</xsl:when>
</xsl:choose>
</xsl:template>

</xsl:stylesheet>

```

Figure 3.10: Extensible Stylesheet Language Transformations (XSLT)

methods for logging information are described here.

- **ServiceOnline():** Publishes system and Service Provider's information and updates the information in the database which is further updated in Application Server. It shows that the Service Provider is available to volunteer services and become part of the Grid. Web Services Description Language (WSDL) is generated when ServiceOnline() web method is invoked. *Provider_Id* and ID are passed as parameters to the Web Service.
- **ServiceOffline():** This web method deletes system information to indicate Service Provider has been shutdown or is not willing to share its information.

System name is provided as parameter to this web method. WSDL is generated when ServiceOffline() web method is called is shown in Figure 3.8.

Authenticated Consumers can retrieve list of resources available on system connected to Grid from status database by specifying their requirements. The Linux platforms can easily be integrated using Mono Tools for other Linux flavors such as Suse Linux 10. In order to use a Web Service, Mono requires a descriptor file written in Web Services Description Language (WSDL). WSDL is an XML description of the API for a given Web Service. Mono's tool wsdl knows how to translate a WSDL file into a *C#* library that can run that Web Service. The local library file is a wrapper around the Web Service, but it gets installed into the Global Assembly Cache (GAC) just like any other library. The Web Services can be consumed by creating a local library that will invoke Web Service using the WSDL file. The Web Service can also be consumed with a Console or *Gtk#* application. These independent nodes using a *ProviderService* Web Service created Thapar University Campus Grid, which was further integrated with Globus and Unicore nodes.

- (c) The open source Globus Toolkit 4 (GT4) contains services, programming libraries and development tools designed for building Grid-based applications. It was developed by the Globus Alliance and other contributors from around the world. The main idea behind the Globus Toolkit is to provide a fundamental and robust infrastructure, tools and libraries for creating a Grid system. The toolkit also provides services commonly used by Grid based applications. The GT4 architecture contains three key components. A set of infrastructure service implementations such as execution management, data access and movement, replica management, monitoring and discovery and credential management (Most of these services are Web Services written in Java). Three containers that can be used to host user-

developed Grid services written in Java, Python and C. A set of client libraries that are used to interact with and invoke GT4 services as well as user-developed Grid services.

The complete version of GT4 was installed on a Linux nodes. Extensive documentation in the form of a quick start guide and a detailed administrator's guide is provided on the Globus Web site. The installation of GT4 was straightforward, it required simply following the instructions in the online documentation.

- (d) Uniform Interface to Computing Resources (UNICORE) offers a ready-to-run Grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet. A UNICORE 6 cluster was developed for implementation purposes where all the UNICORE 6 components such as Gateway, UNICORE/X, Registry, Perl TSI, XUADB (has an associated hostname and port setting) were implemented in a cluster.

3.2.3.2 Translating resource information

The aim of information scheme is to define an information model and map it to concrete schemas to coherently represent Grid resources at different locations and to enable a large multi-locational Grid testbed. The GLUE Schema defines naming standard for Grid Information Services (GIS) which offer varied functionalities and are geographically distributed. The specification describes conventions and terminology related to Grid networks as well as the possible types of attributes and their possible ranges of values for each given Grid property. The GLUE Schema focuses, among others, on:

- sites and services

- computing resources (computing elements, clusters, subclusters), storage resources (storage elements, storage areas, access protocols, control protocols, storage libraries)
- computing and storage domain relationships

Finally, the GLUE Schema stands as a common framework for describing semantic information of stored resources in Grids. The interface between middleware-specific information services and the TUX-INTERO information database is implemented as a five step process: extract, parse, check, transform and load. This process of extracting data from Windows and Linux flavors (acting as one), Common Information Service (CIS) and Monitoring and Discovery System (MDS4), the three existing information systems in TUX-INTERO, transforming this data to the GLUE 2.0 schema and loading it into the target database is realized by middleware-specific gateways, one per middleware. The extractor components of the gateways operate with the standard client APIs of the middleware. The data transformation is implemented using Extensible Stylesheet Language Transformations (XSLT) (as shown in Figure 3.10), which is a popular language for processing XML data or transforming XML documents into other formats. This choice presented itself as most of the information services provide their data in XML format. The data is augmented with the identifiers of the information provider component to keep track of data provenance. In the following, key aspects of the individual gateways implementations, are highlighted.

The Windows Integrated System is designed using Web Services which submits resource information in form of XML and can be transformed easily.

CIS, the information system of the UNICORE 6 middleware, uses the GLUE 2.0 schema internally. Thus, the XSL Transformations for the UNICORE gateway merely checks the XML retrieved from the CIS and maps it to the respective SQL-Commands. No

special rearrangement or interpretation of the data is needed.

One of the most common monitoring systems used within the context of Globus Toolkit/Grid Information Services (GIS) is Monitoring and Discovering Services (MDS), an extensible framework with a hierarchical structure for management of static and dynamic information about the status of Grid components generated by information providers. MDS Index Services provide an aggregation service of lower-level data using a special registration protocol and caching to minimize transfer of non-stale data. The newest release of MDS is built according to the Web Service Resource Framework (WSRF) proposed by OASIS. MDS4 can be queried using a WSRF interface which provides output in XML format. MDS4 makes heavy use of XML and Web Service interfaces to simplify the tasks of registering information sources and locating and accessing information of interest. In particular, all information collected by aggregator services is maintained as XML and can be queried via Xpath queries (as well as other Web Services mechanisms). MDS provides access to static and dynamic information of resources. As MDS4 is based on GLUE 1.1 and additional information providers like GeoMaint [229] provide site related information in GLUE 1.3, the XSLTs of the MDS4 gateway are implemented to parse, check and map the fields with respect to the different versions of GLUE. Globus information can also be extracted using WebMDS. WebMDS is a web-based interface to WSRF resource property information that can be used as a user-friendly front-end to the index service. It provides a simple XSLT-transform based visual interface to the data. WebMDS enables end users to view monitoring information via a standard web browser interface, without installing any additional software on their PC. WebMDS is implemented as a servlet that uses a plugin interface to gather monitoring information (or any other information in XML format) and XSLT transforms and present the data to the user in a readable form. Web

site administrators can customize their own WebMDS deployments by using HTML form options, configuring different plugins to collect data and XSLT transforms and creating their own plugins and XSLT transforms. The loader components of the implemented gateways store the transformed, homogeneous monitoring information in the TUX-INTERO integrated database.

3.2.3.3 Storing resource information

The information regarding the resources is stored in SQL database using Tables in TUX-DB. This information is represented to an end user in a user friendly manner using user interfaces.

3.2.3.4 Querying the resources available

The resource information can be queried using SQL syntax. However, OGSA-DAI was also implemented. OGSA-DAI has a document-oriented interface implemented using Web Services conforming to the Web Services Interoperability (WS-I) basic profile and the Web Services Resource Framework (WSRF). The WSRF implementation of OGSA-DAI is compatible with GT4 and enables stateful Web Services, an important OGSA requirement. The WSRF-compliant version of OGSA-DAI is used because it is compatible with the Globus Toolkit's implementation of WSRF. Having installed GT4, then OGSA-DAI WSRF was installed.

3.2.3.5 Accessing the resources available

The resources can be accessed by using a set of credentials as issued by TUX-INTERO administrator. Chapter 5 shows the experimental results of TUX-INTERO where an end user or Service Requester can login and use the set of resources available on the

basis of trust threshold, discussed in detail in forthcoming chapters.

TUX-INTERO achieves the complex task of integrating resources from various platforms and provide a single massive infrastructure to an end user. Ganglia was chosen as a tool for monitoring the resources in the integrated Grid. Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures. This infrastructure is further secured by using TUX-TMS (proposed trust management solution), as explained in next chapter. Designing a Web Services based solution and integrating resources lead to the next level of integrating more grid resources and making them available through a grid portal due to scalability needs. It lead to the design and development of TUX-INTEROP as a portal for accessing grid resources. The next section discusses TUX-INTEROP in detail.

3.3 TUX-INTEROP: A Portal for secure interoperation of grids

TUX-INTEROP is a grid portal that provides seamless access to a variety of backend grid resources. It provides end users with a customized view of hardware and software resources specific to their particular problem domain. It also provides a single point of access to grid based resources that they have been authorized to use which in turn allow

scientists to focus on their problem area by making the grid a transparent extension of their desktop computing.

3.3.1 Taxonomy of TUX-INTEROP

Following taxonomy has been used in context of TUX-INTEROP:

- (i) Portal: A portal is a web-based application that is customizable by the end-user, both in the look and feel of the portal and in the available content and applications which the portal contains. A portal, furthermore, is an aggregator of content and applications or a single point of entry to a set of tools and/or applications defined by the user or by the portal service provider.
- (ii) Gateway: A gateway is a portal that is dedicated to providing services to a specific user community. When the gateway serves a science community (physicists, chemists, etc.), it is called a “Science Gateway”.
- (iii) Servlets: Servlets are modular Java programs that perform a specific action and create HTML output (akin to Common Gateway Interface (CGI) scripts).
- (iv) Portlets: Portlets are standards-based, portable, special servlets that produce fragments of markup HTML (Hyper text Markup Language) code that can be aggregated into a portal page. Created using an extended Java portlet API, portlets are strung together to build a complex, complete portal. Each portlet typically performs one or more simple actions. However, portlets can also be arbitrarily complex. Portlet standards are determined via the Sun Java Community Process (JCP).
- (v) Web Container: A web container (WC) is a server that runs Java servlets and allows access (to servlets) via HTTP. The most widely used WC is Apache Tom-

cat. Tomcat can run as a stand-alone application and listen for HTTP requests on port 80 or work in congruence with a real web server (such as Apache httpd; communication between Tomcat and Apache occurs via a “connector” in this case).

- (vi) Portlet Container: A portlet container is software running as a Java application inside a web container that allows a portlet to run. In simpler terms, a portlet uses a portlet container which in turn uses a web container. Portlet containers are typically also standards-based, meaning they can run standards-based portlets.
- (vii) JSR-168: JSR 168 is a Java portlet specification standard developed by the Java Communication Process. It defines a “contract” between the portlet container and portlets. JSR stands for “Java Specification Request”. It’s a standard way to “request” additional functionality. A portlet container that is itself JSR 168 compliant can host these JSR-168 complaint portlets without modifications. This is what makes portlets truly portable (the “port” part of the portlet); they can be exchanged easily among JSR 168 portlet containers.

The architecture of TUX-INTEROP is discussed in next section.

3.3.2 Architecture of TUX-INTEROP

TUX-INTEROP is a specialized grid portal that provides an entry point both to the users and grid resources, services and data. The grid portal manages the identity of all users and offers web interfaces for displaying resource information, job scheduling and file/data transfer.

The high level architecture of TUX-INTEROP is shown in Figure 3.11. It identifies the components that make up TUX-INTEROP and their interactions with each other.

The main components are user interface, credential exchange, resource brokering and job monitoring. These high-level interactions are described below:

- (i) The Service Requesters begins interaction by accessing the TUX-INTEROP portal web interface and initiating a request.
- (ii) Resource information is queried by the TUX-INTEROP Portal via the Information services of Service Provider interface and formatted as a response to the clients browser. The role of Information Services Interface is to provide access to the services located at various disparate grids following Service Oriented Architecture (SOA). All resources are defined as services that are accessed by the service consumers layers of the application.
- (iii) In parallel and independently of the rest of the system, the information services take care of collecting updated resource information from the Grid and publishing that through the Service Provider Interface. This updated information then becomes immediately available for display to users.
- (iv) The resources requested are mapped for job execution.
- (v) After a job is completed, the service requester and provider are requested to submit feedback which further calculates trustworthiness and reputation of domains interacting.

The key functions of the TUX-INTEROP are (a) querying for information from various information services, (b) storing and retrieving resource information, and (c) presentation of resource information which is achieved by various layers.

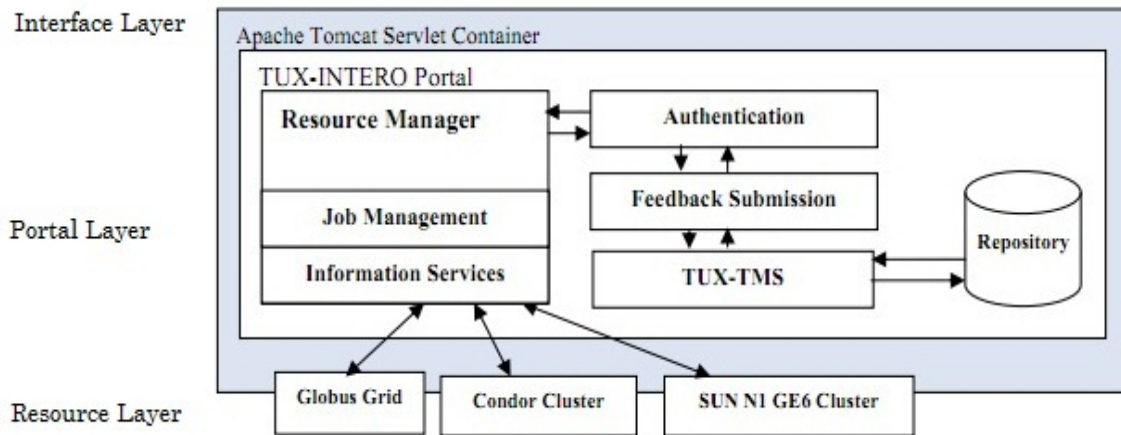


Figure 3.11: Architecture of TUX-INTEROP Portal

3.3.2.1 Resource Layer

Information collection is the responsibility of the Resource Manager (RM). Its role is to gather information from each of the various information services in the system and to publish that information for use by other components. A plug-in architecture has been adopted so that custom adapters can be created and plugged into the system to enable the gathering of information from any type of information provider by making use of Hawkeye and Ganglia for discovery and monitoring. The information is used by querying indexes at the top which are in the form of trees. Though, a limitation encountered is that sometimes due to some failure, latest information may not be available.

3.3.2.2 Portal Layer

The idea is to maintain a cache of the resources which are available instead of storing all the available resources. MS-SQL database is used to store the information. Portal layer has following portlets functionality.

Authentication: A user logs in to the portal with a username and password, us-

ing a standard web-browser. New Users can request an account by registering with valid authentication credential(s). Then, the users will log onto a portal using a web browser and they will authenticate by means of a user-id and password. Once the user is authenticated to the portal server, it is the job of the portal server to act as the user's proxy in most grid interactions. Consequently, the portal server must obtain a proxy certificate that it can use on behalf of the user. The standard approach is to have the user submit a proxy to a MyProxy server. The portal server then retrieves the proxy from the MyProxy server and holds it for the duration of the users session. This process of making the user manage a key pair and submit proxy certificates to a MyProxy server is extremely unpopular with users. Once the user's authenticity is established, access to Grid resources is delegated on their behalf, by uploading a Grid credential or by configuring the portal to access a credential repository.

Resources Selection and Job Execution: Ideally, the user should submit a requirement of resources to the grid. The users should be allowed to access only the resources that have been selected from the registry as shown in Figure 3.12 by applying a set threshold limit based on some parameters such as time, cost, security etc. The portal backend then creates the individual jobs from the user description and maps them to resources.

Thus, from the above requirements it can be seen that special attention should be paid to the following aspects of portal development.

Security: Since the user delegates authority to the portal and the portal potentially has many users, security is an important concern. Ideally users authentication credentials will not be stored on disk of the web server but, rather on a different machine with extra security. All connections to the web server should be made via Secure Sockets Layer (SSL) to protect usernames and passwords. The portal itself should run with

special permissions to reduce damage if it is compromised. Portal user accounts need to only relate to the portal and not to system accounts. The usual drawback of enhanced security is in reduced ease-of-usage.

Resource Discovery: With the plug-in architecture, the Resource Manager (RM) can query various types of information sources, including indexes in Grid middleware such as Globus 4 Toolkit [71] and classads in Condor [22] etc. wherein the repositories may be in the form of SQL database, XML files and Web Services. Therefore, it is able to support different protocols simultaneously, in order for it to gather information from these diverse information sources.

3.3.2.3 Interface Layer

The TUX-INTEROP Resource Manager component provides the user interface for all the resources to be plugged in grid. Using portlet technology, grid resource information is presented to the user via their web browser. The next section discusses implementation details of TUX-INTEROP.

The next section elucidates the implementation details of TUX-INTEROP.

3.3.3 Implementation details of TUX-INTEROP

The overall design of TUX-INTEROP is aimed towards enabling a representation of dynamic and heterogeneous resources by providing enough support to different information providers and allowing easy integration of new source types. To develop the portal, JSR-168 portlets technology was chosen, which ensures flexibility, modularity and extensibility. Portlets are managed by a portlet container which provides a runtime environment to the portlets and manages their lifecycle. A portlet is a Java Web component that generates dynamic content in response to some requests and is

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
Resources.xml contains initial resources when Grid Portlets
starts up.
-->
<grid-resources>
  <hardware-resource label="TIET GridLab Central"
    description="Hosts the GridLab Index Information Services"
    hostname="tiet.thapar.edu">
    <igiis-resource label="iGIIS"
      description="iGrid Index Information Service"/>
    <giis-resource label="GIIS"
      description="Grid Index Information Service"
      basedn="Mds-Vo-name=gridlab,o=grid"/>
  </hardware-resource>
  <hardware-resource label="GridLab MyProxy"
    description="Hosts the GridLab MyProxy Credential Repository"
    hostname="myproxy.gridlab.org">
    <myproxy-resource label="MyProxy"
      description="Online Credential Repository"
      portalProxyFile="/tmp/proxy.pem"
      portalCertFile=""
      portalKeyFile=""/>
  </hardware-resource>
  <hardware-resource label="LMT"
    description="Front-end to the LMT Cluster"
    hostname="lmt.thapar.edu">
    <gris-resource label="GRIS"
      description="Grid Resource Information Service"/>
    <gram-resource label="Globus Gatekeeper"
      description="Globus Resource Management Service"/>
    <gridftp-resource label="Grid Ftp"
      description="Grid Ftp Service"/>
  </hardware-resource>
  <hardware-resource label="RND"
    description="Front-end to the RND Cluster"
    hostname="rnd.thapar.edu">
    <gris-resource label="GRIS"
      description="Grid Resource Information Service"/>
    <gram-resource label="Globus Gatekeeper"
      description="Globus Resource Management Service"/>
    <gridftp-resource label="Grid Ftp"
      description="Grid Ftp Service"/>
    <grms-resource label="GRMS"
      description="GridLab Resource Management Service"
      protocol="http"
      port="8543"
      serviceDn="/C=PL/O=GRID/O=PSNC/CN=grms_devel/rnd.thapar.edu"
      servicepath="/axis/services/grms-testing"/>
  </hardware-resource>
</grid-resources>

```

Figure 3.12: Resource Description in TUX-INTEROP

typically displayed as a small window inside of a Web page. The web interface of TUX-INTEROP is implemented using a set of portlets. GridSphere version 3.1 has been used as the portlet container for implementing the portlets. Apache Tomcat is used as the servlet container, provides the necessary HTTP, HTTPS for communicating with the portal using a web browser. GridSphere is used as the portal framework, by which developers can easily create portlets conforming to the JSR- 168 standard.

GridSphere provides some generic portlets and portal layout management. JSR-168 compliant portlets are used to implement the TUX-INTEROP portal as a portable and pluggable component which can be integrated into other portals. The components are kept as modular and general purpose as possible to make the system easy to extend and adapt to, various scenarios. The TUX-INTEROP portal is developed based on JSR-168 compliant portlets. TUX-INTEROP Portal Portlets offer functionalities such as user authentication, job submission, resource discovery and feedback submission. In this section, technologies chosen to achieve this goal have been outlined.

3.3.3.1 Grid Middleware

The Globus Toolkit (GT) provides functionality to glue together distributed resources to form computational grids and to create grid based applications. It provides a secure, uniform interface for resource access and provides file transfer, remote job submission and meta-data query services and is one of the most widely-used middleware packages for the grid. GT is increasingly being developed in Java, which is also well suited for building complex web applications. Java-based frameworks were used for structured development and ease of use of the Globus Toolkit.

3.3.3.2 Local Resource Management Systems (LRMS)

SUN NI GE 6 and Condor have been chosen as Local Resource Management System (LRMS) to be connected to portal for providing resources to the end users.

3.3.3.3 Web Technologies

Servlets are a viable choice for reusable web applications and a number of different frameworks built on top of servlets were explored during the course of development.

Portlets provide a standard technique for creating web applications for deployment in a container portal. Each application is self-contained and is displayed in a subwindow of the portal. The portlet framework allows for communication between portlets without the need for tight inter-portlet coupling. Java Specification Request (JSR) 168 defines the standard for portlets and several systems implement it, including Apache Jetspeed2 and GridSphere. GridSphere was chosen as a preferred portlet container. It is based on the popular Apache Tomcat5 web-application container. The ApacheAnt4 build system, which is used by GridSphere, allowed to semi-automate this process.

GridSphere was chosen as the portal framework as it complies with the needs of the Grid Community. GridSphere provides a sophisticated Portlet Service Mode that encapsulates the reusable portlet logic into services that may be shared between many portlets. The first versions of GridSphere have provided support for two sets of portlets: Open Grid Computing Environment (OGCE) and GridPortlets, the second becoming a part of OGCE [231]. The first versions of GridSphere have provided support for two sets of portlets: OGCE and GridPortlets, the second becoming a part of OGCE. TUX-INTEROP was developed using GridPortlets. GridPortlets make it difficult for end users to plug new portlets as and when the functionality and resources changes. As the functionality of Grid portals grow and more information is supported, ease of use becomes more important. End-users from different working area focus on different information sources and resources. So, a well designed grid portal should support personalization which was lacking in TUX-INTEROP as GridPortlets became outdated and cumbersome to work on for every end user. The aim of allowing each user to be able to design his own personal space and put useful information on his pages and also edit and update this personnel space in future propelled to propose Pluggable, Interoperable and Extensible (PIE) grid portal with latest portlets available for deploying

and using grid. The next section discusses PIE's architecture and implementation in detail. The architecture of PIE is motivated by that of TUX-INTEROP.

3.4 PIE: A Grid Portal for Interoperability

Pluggable, Interoperable and Extensible (PIE) is a grid portal which can be used by scientists and end users to easily run their jobs on Grid Computing infrastructure. Scientists can easily develop their portlets and plug-in to PIE. It uses portal based solution to allow an end user access to heterogeneous resources available in Grid Computing Environments. End users can easily plug-in heterogeneous resource without building the gateway. Grid-specific functionality is accomplished with Vine portlets that make use of the Java CoG Kit to actually carry out Grid-related operations. Vine portlets include: Account Manager, Credential Manager, File Manager, Job Manager and Resource Manager. The next section unveils taxonomy and architecture of PIE.

3.4.1 Taxonomy of PIE

The taxonomy of PIE remains same as that used for TUX-INTEROP.

3.4.2 Architecture of PIE

PIE system architecture is shown in Figure 3.14. PIE architecture is divided into three layers structure: resource layer, interoperability layer and portal layer.

3.4.2.1 Resource Layer

The lowest layer in PIE architecture is the resource layer. It hosts various heterogeneous resources as available in a grid. The resources may be running on different middleware

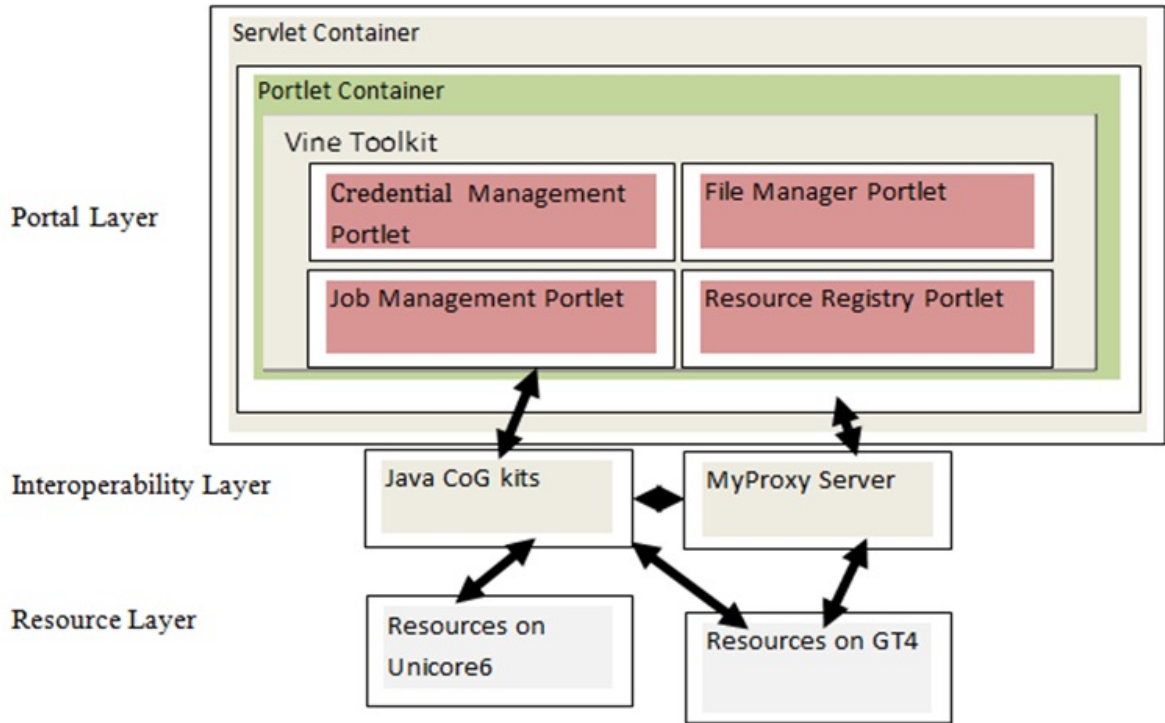


Figure 3.13: Architecture of PIE

in a form of cluster of systems which is plugged in to provide services to the Grid users.

3.4.2.2 Interoperability Layer

The intermediate layer is the interoperability layer. It hosts the portlets which have Java CoG (Community Grid) kits at the backend to serve as gateway to the resources underlying. Commodity Grid (CoG) Kits allow grid users, grid application developers and grid administrators to use, program and administer grids from a higher-level framework. It helps the developers to bridge between even different version of same middleware such as GT2 and GT4 of Globus Toolkit. It also has Certificate Manager, which stores credentials and maintains them. The Certificate Manager used in PIE is installed from Vine Toolkit which enables an administrator to maintain credentials for each user. The credentials maintained are from multiple middleware and are composed

in multiple formats. A user can only submit job to the resource only if he has the required credential to access the resource. It becomes insecure to store credentials on a user's desktop. Therefore, a central repository of credentials enables easier and efficient maintenance and access to resources. The access rights are available to the user on the basis of the credibility of the credentials available.

3.4.2.3 Portal Layer

The topmost layer is the portal layer. It includes portal framework. The grid portlets are hosted in portal framework and managed by portlet container. When end-users login in the user page, the portlets will visit the resource registry, retrieve resource information and display back on the portlets. So, the end-users can get the resource information and decide to run their jobs on the resources available. The Portal is further hosted in servlet container. The Portal layer consists of Portlets responsible for various primary functions as accomplished by Portlets. The Portlets are Credential Management, File Management, Job Management, Security and Resource Management. The portlets are discussed here briefly:

Credential Management: The Credential Management Portlet enables user to manage and upload credentials required for running a particular job. New credentials can be stored and used. It works separately from the Certificate Manager.

File Management: The File Management Portlet enables user to store and maintain data as needed as it forms the single most important asset in a Grid Computing system. This data may be input into the resource and the results from the resource on the execution of a specific task. If the infrastructure is not designed properly, the data movement in a geographically distributed system can quickly cause scalability problems. The data is expected to be near to the computation where it is used. The

File Management Portlet enables easier access to data files.

Job Management: The Job Management Portlet enables an end user to submit the job in a user friendly way by selecting the resources. The job is converted into Job Service Description Language (JSDL) which is a standard format to submit job on grid. The job management service for users provides job allocation, status management, data distribution and start/stop jobs. JSDL specifies different information required for successful application execution. One of the sub elements is application element which allows the user to specify the application to be executed on the grid and the various application parameters and arguments, including input and output files. Requirements element allows the user to specify different requirements needed for the execution of the job, such as amount of memory, number of CPUs, etc. Data element allows the specification of input and output staging, in order for the user to optionally define which files should be copied to and / or from the working directory on the host where the job will be executed before job submission and after job completion, respectively.

Security: The resource access and information is available to only authenticated users. The users are supposed to produce credentials to access resources which can be of multiple mechanisms such as MyProxy, VOMS etc. The heterogeneous nature of resources and their differing security policies are complicated and complex in the security schemes of a Grid Computing Environment. These computing resources are hosted in differing security domains and heterogeneous platforms. The Grid Computing data exchange must be protected using secure communication channels.

Resource Management: The resource registry portlet configures the resources available in the grid. The resources are configured usually using domain.xml file. The most interesting aspect of the resource management area is the selection of the correct resource from the grid resource pool based on the service-level requirements and then to

efficiently provision them to facilitate user needs.

These portlets enable an end user to use grid infrastructure easily without having the need to bother about developing the grid infrastructure and managing separate credentials for different resources.

```
- <domain name="vo.thapar.edu" label="THAPAR Grid" description="Thapar Grid">
- <!-- Portlet authentication -->
- <authenticationModule key="PortletAuthModule" order="1" />
- <!-- Credential repository authentication used with MyProxy server -->
- <!-- authenticationModule key="CredentialRepositoryAuthModule" order="2"/ -->
- <!-- Proxy created from local user cert and key -->
- <!-- authenticationModule key="VomsDefaultCredentialAuthModule" order="3"/ -->
- </vineRoleResource />
- <!-- Portal -->
- <hostResource name="portal" hostname="localhost" label="Portal" description="Portal">
- <!-- Portal file system -->
- <portalFileSystem label="Portal File System" description="Portal File System" />
- <!-- Account manager -->
- <accountResource name="GuestAccountManager" label="Guest Account" description="Guest Account Manager">
- <gridsphereRegistrationResource name="GridsphereRegistration" label="GridsphereRegistration" />
- </accountResource>
- </hostResource>
- <!-- MyProxy resource LMT Thapar -->
- <hostResource name="server" hostname="lmt.thapar.edu" label="server" description="Linux Host (2 CPUs)">
- <!-- MyProxy -->
- <myproxyResource label="knoppix (MyProxy)" useCredential="false" checkConnection="true"
timeoutMilliseconds="5000" />
- </hostResource>
- <!-- seagrass test uncore -->
- <hostResource name="gridlab" hostname="gridlab" label="gridlab" description="Linux Host (2 CPUs)">
- <!-- Unicore Gateway -->
- <unicoreGatewayResource name="unicore" label="gridlab (UNICORE6)" port="7777" protocol="https"
servicePath="/DEMO- SITE/services">
- <resourceAttribute name="description" value="UNICORE Gateway 6.0 version 1.0" />
- <resourceAttribute name="serviceTimeout" value="3" />
- <resourceAttribute name="userKeystore" value="true" />
- </unicoreGatewayResource>
- <!-- Unicore SMS -->
- <hostResource name="icmWMS" hostname="coegc.thapar.edu" label="COEGC" description="Resource broker in
ICM">
- </hostResource>
- </domain>
```

Figure 3.14: Domain.xml of PIE

3.4.3 Implementation details of PIE

In a grid, managing resources, services and computations is a challenging due to the heterogeneous nature, large number, dynamic behavior and geographical distribution of resources. This presents the need to easily provision the resources and enable in-

teroperability between the resources. PIE has been deployed on various Pentium Core 2 Duo Machines running Windows Xp and Ubuntu as Operating System and GT4 and UNICORE 6 as middleware. Gridsphere 3.1 was considered for Portlets container and Apache Tomcat-5.5.26 was considered for Servlet container. Vine toolkit 1.2 was deployed which run Java CoG kits at the backend to connect to the clusters of Globus Toolkit and Unicore 6. In this section, technologies chosen to achieve this goal have been outlined.

3.4.3.1 Grid Middleware

At the resource layer clusters of Globus and Unicore were installed. The Globus Toolkit (GT) provides functionality to glue together distributed resources to form computational grids and to create grid based applications. It provides a secure, uniform interface for resource access and provides file transfer, remote job submission and meta-data query services. It is also one of the most widely-used middleware packages for the grid. GT is increasingly being developed in Java, which is also well suited for building complex web applications. Java-based frameworks for structured development and the ease of use were chosen to develop. GT4.2.1 version was chosen to be installed.

Uniform Interface to Computing Resources (UNICORE) has been chosen as another grid middleware where UNICORE 6 Servers were installed. UNICORE offers a ready-to-run grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet. Unicore-servers-6.3.2 was installed to enhancing pool of resources.

3.4.3.2 Web Technologies

Servlets are a viable choice for reusable web applications and a number of different frameworks built on the top of servlets were explored during the course of development. Apache Tomcat-5.5.26 was chosen as web-application container.

3.4.3.3 Portal Container

The GridSphere portal framework provides an open-source portlet based web portal. GridSphere enables developers to quickly develop and package third-party portlet web applications that can be run and administered within the GridSphere portlet container. GridSphere was chosen as a preferred portlet container. The Apache Ant-1.8.2 build system, which is used by GridSphere, allowed to semi-automate this process. GridSphere-3.1. was chose as portal container. In addition to being portal container, the GridSphere version 3.1 distribution includes a collection of basic user/administrative portlets: portlets for users to login/logout, for manage profile information, for starting, stopping or redeploying portlets, as needed.

3.4.3.4 Vine Toolkit

Version 3 of GridSphere is not 100% compatible with OGCE portlets and recommends instead Vine portlets that are in development, unlike OGCE project, that has been frozen, so that the reason for this choice is clear. Vine is a modular, extensible Java library that offers developers an easy-to-use, high-level Application Programmer Interface (API) for grid enabling applications. Vine Toolkit Version 1.2 was installed for embedding portlets such Credential Management, File Management, Job Management and Security for grids. Vine Toolkit is advancement over GridPortlets and OGCE, lately. The portlets can be designed in Adobe Flex and easily embedded into PIE.

The developers can easily configure registry portlet to plug-in the resources for further usage purpose and extensibility of portal.

To install Vine portlets, a Web container such as Apache Tomcat-5.5.26 (or newer) and a GridSphere 3.1 portlet container is installed and configured as needed. Vine portlets come with configuration file `domain.xml`. located in `$CATALINA_HOME/webapps/vine/WEB-INF/vine/resources`. The `domain.xml` file has been configured for the resources available at Centre of Excellence in Grid Computing at Thapar University. The domain name depicts the domain related information of Thapar University. The `< authenticationModule >` in `domain.xml` file configures the type of authentication mechanism used to allow users to register to the Portal and make use of resources. The logic responsible for the user authentication is that the basic authentication module “PortletAuthModule” relies on the GridSphere authentication routines and checks whether user was successfully authenticated by GridSphere. The Portal File System tag is responsible for configuring the file system to be deployed. The `< accountResource >` is responsible for user’s management. It manages user’s roles and access rights to the underlying resources. The `< hostResource >` acts as a container or parent resource. It conforms to the physical host machine with some services installed on it. The `< UnicoreGatewayResource >` provides information related to the resource which has Unicore6 Server installed.

TUX-INTERO, TUX-INTEROP and PIE have been discussed here as solutions to interoperability challenges in a grid. Table 3.1 compares TUX-TMS with existing trust management solutions. The testing results have been demonstrated in Chapter 5.

Table 3.1: Comparing existing Interoperability solutions with Proposed solutions

Types of Solutions	Features	Existing Solutions	Proposed Solutions
Adapters	Interfaces resources running one of these infrastructures: GT2, GT4, and LCG.	GridWay	TUX-INTERO: Interfaces resources running on GT4 and Unicore and are extensible
Web based Portals	Collaborative access to high-performance parallel and distributed applications for interaction and steering	DISCOVER	TUX-INTEROP and PIE are portal based solutions which are extensible
Grid resource descriptions	describing abstract spaces between which quantitative descriptions of requests for computational resource must be mapped to resource provision	Brooke	Web Services used in TUX-INTERO for providing resource description using SIGAR
Grid workflow management system	Performance prediction and Grid resource management/Job/service orchestration and execution	Gridflow, P-Grade, Pegasus	-
Transatlantic testbed	Common schema for resource location and status in the context of a Virtual Organization	WorldGrid	Translates varied schema into a common schema
Broker	Interoperates with various batch systems as well as multiple Grid middleware	Venugopal[103], Kertesz[104]	-
Metabroker	Unified mechanisms for all tasks in the job management process and interact with Grid-specific brokers	Rodero[105]	-
Metascheduling	Architectures for interoperable metascheduling and present a hybrid approach that combines hierarchical and peer-to-peer architectures	Bobroff[106]	-
WSRF-based MetaGrid	WSRF-based MetaGrid services as a bridge between users and different Grid	Peirantoni[107]	-
Standardisation	Developing standards for interoperability	Genesis II[108], OGSA standards[109], ETSI[115], OMII[116], UniGrids[118], Grimme[119], Daniel[120], LHC[121], Elmroth[122], Timo[124], GIN[144]	Makes use of already standardised models for enhanced usage such as XSLT, OGSA-BES etc.
Delegated match-making architecture	Hybrid hierarchical/distributed architecture	DMM[127]	Resources are added as per the availability

3.5 Conclusions

To conclude, this chapter explains taxonomy, architecture and implementation details of the proposed interoperability solutions: TUX-INTERO, TUX-INTEROP and PIE. TUX- INTERO is a Web Services based Interoperability System which integrates resources on Windows and Linux Platforms using Web Services designed in .Net Platform and thereafter, couples with clusters on different grid middleware. TUX-INTEROP is a grid portal that provides seamless access to a variety of backend grid resources. It provides end users with a customized view of hardware and software resources specific to their particular problem domain. It also provides a single point of access to grid based resources that they have been authorized to use which in turn allow scientists to focus on their problem area by making the grid a transparent extension of their desktop computing. Finally, PIE which is a pluggable, interoperable and extensible grid portal is discussed which can be used by scientists and end users to easily run their jobs on grid computing infrastructure. The most advantageous feature of PIE is that Scientists can easily develop their portlets and plug-in to PIE which is usually not available in grid portals.

Chapter 4

The Proposed Trust Management Solution: TUX-TMS

Based on the Security challenges discussed in previous chapters, this chapter proposes TUX-TMS (Thapar University Extensible - Trust Management System), a solution to the trust management challenge in Grid Computing Environments. This chapter presents taxonomy, architecture, design and implementation details of TUX-TMS. The evaluation and other trust parameters deemed essential for TUX-TMS are also elaborated and discussed in detail. For enabling secure transactions across Grid, TUX-TMS incorporates various components and ensures reliable information to the an entity on the trustworthiness and reputation information of unknown domains.

4.1 Trust Management

Trust management mechanisms are a promising solution for strengthening the confidence in the quality of the interaction process between Grid participants. Trust in Grid Computing Environments can be defined as the extent to which every participant in a

Grid environment, in a specific moment of time, with an evidence of relative security regarding the identity and the behavior of their counterparts, is willing to interact with them, even though unexpected negative outcomes could result from the entire interaction process. Although, encapsulating trust in Grid Computing will augment transactions and establish secure Grid environments, but trust modeling and interaction in a Grid system is a complicated issue, to address. This scenario not only requires a technical solution, but it also requires regulatory and legislative protections on the transactions themselves. This is a plausible requirement, since it is the standard for off-line commercial transactions, also. The main motivation in studying trust management is to facilitate Service Requester as well as Service Provider to make more informed decisions for utilizing Grid resources. Trust management is the usage of trust information to allow establish secure environments for usage of services amongst Service Requesters and Service Providers. Thus, trust management system involves the act of using trustworthiness and reputation information of entities to incur transactions. These transactions further involve the process of submitting feedback as an evidence which is analyzed to uncover new (and/or unwanted) relationships or side-effects and presenting information to the user that can be used to make trust decisions. This evidence is collected from the source of the interactions. Thus, trust management also involves the monitoring and (re)-evaluation of the entity's trust information. This perspective is the view taken by the Thapar University Extensible - Trust Management System (TUX-TMS), proposed trust management solution. The next section discusses TUX-TMS in detail by circumventing from taxonomy, parameters, architecture to implementation details.

4.2 TUX-TMS: A flexible reputation based trust management solution

TUX-TMS is a flexible Reputation based Trust Management system which combines the merit of policy proof and reputation evaluation. TUX-TMS takes feedback from the entities and computes trustworthiness and reputation score of the entities and domains to enable decision making for performing the transactions. This section discusses taxonomy and parameters of TUX-TMS.

4.2.1 Taxonomy of TUX-TMS

The taxonomy classifies trust management system by characterizing its different attributes. The taxonomy in this section describes the different aspects of the Reputation based trust management system that provide trustworthiness and reputation information to the end user. TUX-TMS has been proposed for computational Grids. The taxonomy of TUX-TMS is presented here:

- (i) Entity: A user or resource requesting resources or services.
- (ii) Domain: A group of entities which are part of one organization controls access to its resources.
- (iii) Virtual Organization: A group of domains which collaborates under common objective.
- (iv) Service Requester/Resource Consumer: A Service Requester is an entity or domain, which requests some services such as computing, storage etc. from the Service Providers.

- (v) Service Provider/Resource Owner: A domain which provides services to an entity/domain requesting them.
- (vi) Services: Services such as computational, data storage, printing, using software licenses or scientific instruments which may be provided to the Service Requester.
- (vii) Transaction: When a Service Requester uses the services of the Service Provider and pays the amount for the services used and submits feedback for the same, a transaction is assumed to be complete.
- (viii) Service consumed: A service is said to be consumed when a transaction is complete.
- (ix) Service Initiated: Services are initiated when jobs are mapped onto resources.

4.2.2 Trust Evaluation Parameters

An evaluation parameter is a measure of how an entity of a domain is trusted by the other entities. TUX-TMS is a transaction based feedback system, where the feedback is mandatory with each transaction. In TUX-TMS, the trust evaluation parameters derived are:

- (i) The trustworthiness is an indicator of the quality of the entity's or domain services. The higher the value, the higher the trustworthiness of the domain.
- (ii) The reputation of a domain is the expectation of its behavior based on other entities observation or information about the entity's past behavior within a specific context of a given time. The higher the value given by diverse domains, the higher the reputation of the domain.

- (iii) The feedback score that a domain or entity receives from other domains in terms of service provided. This value is average of the feedback score over a number of transactions. The trust relationships assumed for TUX-TMS are indirect, recommendation and direct trust relationship and have been assigned weights between 0-1.
- (iv) The trust and reputation values decay as the time progresses and needs to be established again between the domains after a period of time has lapsed. When the value are below threshold, the trust and reputation values are to be gained again.
- (v) Risk assessment establishes the risk level involved in transacting with a domain. The higher the value, the higher is the risk involved in transacting with the domain.

4.2.3 Trust Parameters

TUX-TMS should cater to various parameters and adhere to various rules and regulations in case of establishing and evaluating trustworthiness and reputation information of domain in Grid Computing Environments. The following parameters have been considered essential for TUX-TMS:

Trustworthiness: Trustworthiness is assumed as firm belief in the competency of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time.

Reputation: The reputation of an entity is an expectation of its behavior based on other entities observation or information about the entity's past behavior within a spe-

cific context of a given time. A online transaction with a prospective new partner is risky if it is not known how he has behaved in the past. So reputation makes transaction predictable and secure.

Identity trust: Identity trust focuses on verifying the authenticity of an entity and determining the authorization that the entity is entitled to access and is based on techniques including encryption, data hiding, digital signatures, authentication protocols and access control mechanisms. An Identity Management System has been deployed in TUX-TMS to verify and validate credentials submitted by the users.

Behavior trust: The behavior of grid participants is unpredictable. Participants can behave either “good” or “bad”. In most cases, “good” behavior reflects the expectations of a participant to simply receive a response from another participant involved in an interaction or sometimes to get accurate results. If an interaction party behaves differently from these “normal” expectations, its behavior is labeled as “bad”. Participants with “good” behavior are considered as trusted ones and are thus eligible for future interactions. Participants with “bad” behavior have only minor or no possibility to be considered for further interactions. Table 4.2 shows partitioning of users on the basis of their behavior.

Feedback: Feedback is submitted on the performance of the entities after each transaction is finished and information is stored for use by future entities. It also provide scores summarizing the history of transactions that an entity has engaged in. In TUX-TMS, after an interaction feedback is taken, according to which trustworthiness and reputation information is updated. The feedback is submitted by the Service Requester as well as the Service Provider for the interacting domain. To discourage malicious users from submitting wrong information, feedback is taken in form of a questionnaire and the scores cannot be made out easily by a malicious user. If the feedback is high

the trust level is raised and correspondingly if the feedback is low, the trust level is lowered.

Trust Context Factor: At any time t , the purpose of using services of a Service Provider by a Service Requester can be termed as the trust context factor, which may vary from application purpose. The trust weights may be assigned and evaluated likewise. In TUX-TMS, Computational and Data service are given higher context and Print service is given lower weight for testing purposes.

Trust and Reputation Updation: Trust and Reputation information is updated after a successful transaction. According to the feedback collected about an interaction, Trust base (TB) and Reputation Base (RB) is updated. A number of solutions exist to protect privacy. First of all, introducing extra latency between the time that ratings are submitted and the time when the new score is published can make timing correlation more difficult. On the other hand, this might reduce the quality of the system, because scores are not updated immediately.

Trust and Reputation Decay: Trust and reputation values decay with time. The positive effect of successful interaction on trust will reduce over time as will the negative effect of unsuccessful interaction. The value of trust today may not have the same significance after, say 3 years. So, a time variable t_0 is associated with the trust value.

Trust Symmetry: Entity A trusts entity B, but does entity B also trusts entity A? The solution to this problem is Trust symmetry. It can only be answered in a non-deterministic manner, as the entity A or user needs to position itself as the resource provider host or entity B to estimate their trust on the use from user's point of view, that is, to evaluate the trust reflection. This is difficult without knowing the resource provider's mind set (i.e. trust in the user), as in this case the user may not have all the hard facts about the resource provider's trust on the user. Current techniques mostly

project only the resource provider from attacks by the user, while leaving the user comparatively dependant on the well behavior of the resource provider. In TUX-TMS, the entities are allowed to access resources on the basis of trust information available of the Service Providers.

Historical accumulation of past behavior: An entity that interacts with another entity, after evaluation of trust, has to store that trust value for future use. A data structure, say, table may be used for accumulation of trust value of an entity for another specific entity. This data structure contains the trust level (which may range from highly untrustworthy to highly trustworthy) of one entity on other as shown in Table 4.1. Entities interact for a particular context, say, printing, storage, computation etc. An entity x that has trust on entity y for a particular context, stores an entry in its data structure. This entry is the trustworthiness score of entity x on entity y and this trust denotes the direct trust of entity x on entity y .

Weightage of identity trust and reputation: An identity for which both direct trust and recommendation trust exist, direct trust is given a higher weightage than the recommendation trust. The trust relationships considered for TUX-TMS are direct, indirect and recommended, which determines the trustworthiness of entities in case of entities sharing same trustworthiness and reputation values. Here, direct trust is given more weightage than recommended trust.

Trust level: Trust can be categorized into levels. Trust can range from ht to hu as shown in Table 4.1 where et is rarely achievable therefore, not considered for real models. The values obtained places the users in the range of trust levels.

Trust Inheritance: In a Grid environment, entities are dynamic, they can join and leave a virtual organization at any time. Hence, a trust model for such an environment should have mechanisms for managing trust for such entities. However, the other do-

Table 4.1: Trust Level

Trust level	Significance
et	Extremely high trustworthy
ht	Highly trustworthy
t	Trustworthy
u	Untrustworthy
hu	Highly untrustworthy

mains might not trust this new entity to be as trustworthy as another entity who has been with interacting for a longer period of time. Therefore, there is a member weight associated with every entity to indicate if the entity is a new, recent, or an old member with its domain and each individual domain can decide what constitutes an entity to fall in one of these member weights. When an entity joins a domain, it inherits the T_{fin} from the domain it is part of and calculated using Equation 4.9.

Evolving trust as a newcomer: As a new entity joins a domain, it initially has no Trustworthiness and reputation values. A newcomer is allowed to interact with domain having certain threshold limit for services and it can then build its own trust values or inherit. When new entities want to join the system and it is not part of any existing domain, then they must contact certain domains that are acting as introducers. These introducers are domains that are willing to advertise their existence in order to introduce new entities in Grid. Introducing consists simply of broadcasting a reputation referral with some initial reputation values.

Trust threshold: Trust threshold may be defined as minimum trust values that are required to access the services. Trust threshold can be defined as the minimum value of trust that is required to establish a trust relationship between a trustor and a trustee for assessing some service. The entities having trust value more than trust threshold qualify for assessing a service.

False Recommendation: Reputation of an entity is a very important factor in evaluating trustworthiness, especially in the case where there is no direct trust or the TB is empty. Reputation of an entity is calculated from recommendation of several other entities. Any entity can give fraudulent recommendation, which is taken care of by aggregating and auditing all the recommendation given by a particular entity.

Transitive Recommendation: If A trusts B and B trusts C then A may also trust C. In open distributed environments, trust information can be collected from recommenders and trust base to check the authenticity of the users and then transaction may be committed.

Risk assessment: For interacting parties in a distributed platform as Grid, there is always an element of risk involved. Generally, risk can be defined as the exposure to uncertainty with a known probability of the distribution of events. The connection between risk and trust is sparsely researched. From general observation, risk and trust are inter-related in following ways:

- Risk may be used to determine the level of trust.
- Trust may be used to determine riskiness of a service.

In both cases, there is an inverse relationship between the two concepts. Risk assessment accounts for the bad behavior of the entity. Even a good known entity may also behave badly in some situations. Risk assessment checks the risk involved in transacting with a particular entity. The ranking of domains in Domain Evaluation Table determines risk involved in transacting with a particular domain by dividing rank of the domain by total number of domains in a Grid, at a particular time t_0 .

Aggregation of all recommendations: Suppose an entity x wants to interact with entity y for a particular context c. Entity x will calculate Trust on the basis of direct

trust and reputation. For calculating reputation, x will ask its set of recommenders for giving recommendations and if any of those recommenders do not have a value in their Reputation Base (RB), they will further ask their set of recommenders for giving recommendations and the process may continue thereby forming a chain of recommendations. These chains form a recommendation network which is solved by both aggregation of recommendations and transitive recommendations.

Intrusion Detection and Audit trail: Intrusion detection is a process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violation or imminent threats of violation of computer security policies. For detecting authorized but abusive user activity, audit trails may be an appropriate means. Audit trails can establish accountability of users for their actions. They can provide evidence to establish the guilt or innocence of suspected individuals. If any malicious activity is found, trust level decreases and penalties can be imposed on the defaulters. Audit trail analysis can be in-depth offline (after the fact) analysis of audit data; real-time testing of audit data, so that an immediate response is possible and subsequent analysis of audit data for damage assessment. The measures used for detecting abnormal activity may be CPU usage, I/O usage, location of use, compiler usage, system call usage, directory and file usage, system errors, remote network activity, network activity and many more. Auditing is also performed considering parameter for false recommendation. The ultimate goal of any Intrusion Detection System (IDS) is to obtain as high detection rate as possible and as low false alarm rate as possible. These are helpful in finding the feedback. If any abnormal activity is found, the malicious domain is penalized by decreasing the trust level. Further, penalties can be imposed on the defaulters in form of fine or blacklisting the defaulters after significant evidence.

Interoperability: Interoperability aims at federating the use of user's credentials and providing a unique authentication and authorization framework. Users logged into disparate middleware are allowed to log in to the system and access Grid.

3-D view: The three dimensional view of trust denotes belief, disbelief and uncertainty. Belief means the probability that an entity B can be trusted by another entity A and disbelief means that B cannot be trusted by A. Then, uncertainty fills the void of absence of both belief and disbelief, sum of these elements should be 1. A mapping function is applied to map two dimensional evidence to three dimensional trust.

After elaborating the parameters for TUX-TMS, the next section presents partitioning of users on the basis of their behavior patterns.

4.2.4 Partitioning of the Users

The users can be categorized into four levels according to their behaviors patterns. The security threats and auditing conducted, places the users into these zones as shown in Table 4.2. The different zones decides the Percent character (P_C) of the user as calculated by Equation 4.1, where n stands for negative transactions, p stands for positive transactions and σ is total number of transactions. (P_C) is decided after minimum 5 transactions and can be determined on the basis of the flow of transactions in a Grid.

White zone: The users who have always submitted honest feedback and there has been no security violation or malicious activity reported against them come under white zone users. It is calculated using Equation 4.1 and users above 75% lying in this zone where positive transactions are greater or equal to total transactions and where negative transactions are presumably negligible.

Blue zone: The users who in spite of having good reputation and who have been

Table 4.2: Partitioning the users

Zone	Trustworthiness	Risk involved
White	High	Low
Blue	Medium high	Medium
Grey	Medium	Medium high
Black	Negligible/Low	Very High

submitting honest feedback in the starting have all of a sudden started submitting false feedback information come under the category of blue zone users. It is calculated using Equation 4.1 by having users above 50% and less than 75% lying in this zone. Here, positive transactions are greater than negative transactions and where negative transactions are presumably low.

Grey zone: The users who have submitted some honest and some false feedback information and who have reported security violations against them come under the category of grey zone users. It is calculated using Equation 4.1 by having users above 25% and less than 50% lying in this zone. Here, negative transactions greater or equal to positive transactions total transactions than the negative transactions and where negative transactions are presumably more.

Black zone: The users who have only submitted false feedback and have only malicious activities come under black zone users. It is calculated using Equation 4.1 by having users above 0% and less than 25%, lie in this zone. Here, negative transactions are greater than the positive transactions and where negative transactions are presumably significant.

These zones are calculated using the equations mentioned here, where P_C stands for Percent character, n stands for negative transactions, p stands for positive transactions and σ is total number of transactions. Here, transactions that are violating Service

Level Agreement (SLA) are considered negative and those which adhere to them are considered positive.

$$P_C = n + p/\sigma \quad (4.1)$$

Further, a phylogenetic tree has been created showing the inferred evolutionary rela-

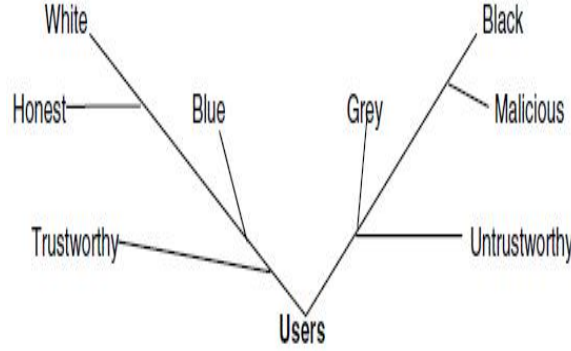


Figure 4.1: Phylogenetic tree of Users in TUX-TMS

tionships among various users in these zones based upon similarities and differences in their identity and behavioral characteristics. An example is depicted in Figure 4.1. In a phylogenetic tree, the taxa joined together in the tree are implied to have descended from a common ancestor. In a rooted phylogenetic tree, each node with descendants represents the inferred most recent common ancestor of the descendants and the edge lengths in some trees may be interpreted as time estimates. Each node is called a taxonomic unit. Internal nodes are generally called hypothetical taxonomic units (HTUs) as they cannot be directly observed. The tree can be further, evolved for a number of users having varied behaviour according to context used for evaluation. Phylogenetic tree has been created to extend this model to incorporate different zones in future according on behavioral patterns of users, which keeps changing.

4.2.5 Architecture of TUX-TMS

The architecture of the TUX-TMS is depicted in Figure 4.2. The lowest layer comprises of resources in the form of clusters and which are located in various administrative domains. The next layer is the Binding Layer which allows various service requester and service Providers to become part of Grid. The topmost layer is the Application Layer which has all the TUX-TMS Components embedded for various trust related functions. These functions are presented as a use case here. An end user who is a Service

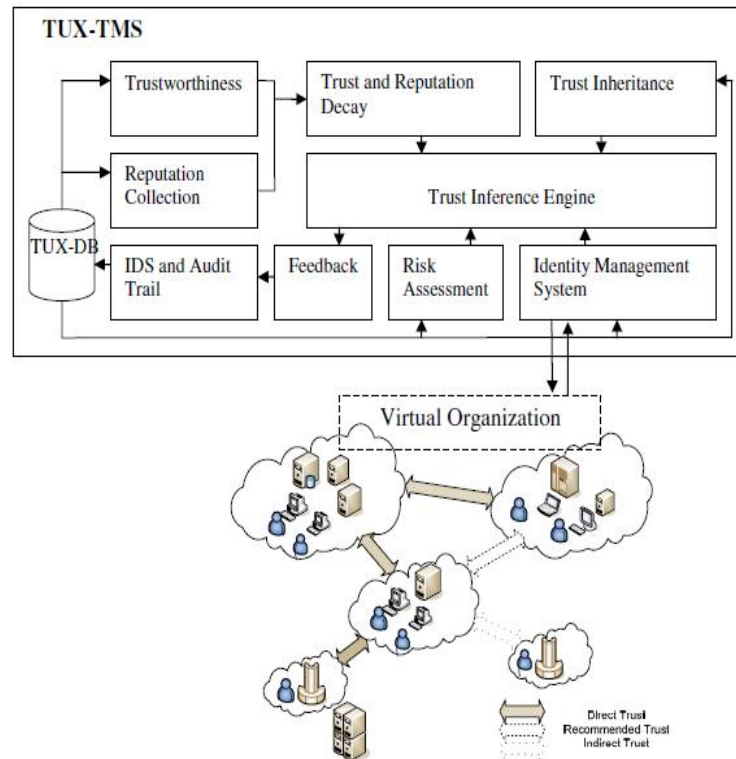


Figure 4.2: Architecture of TUX-TMS

Requester is requested to login into the TUX-TMS using his credentials as shown in Figure 4.3. If the user is already registered with the TUX-TMS as shown in Figure 4.4, the Identity Management System checks and verifies authentication information provided such as security certificates or else if a user is new, he is requested to register

as shown in Figure 4.5. Firstly, the values of Trust Inheritance, Risk Assessment and Trustworthiness (TD) and Recommendation (RD) are checked and then a user is allowed a set $M \{D, Sp, S\}$ where D is a set of domains $\{D_1, D_2, D_3..\}$, and Sp is a set of Service Providers $\{Sp_1, Sp_2, Sp_3..\}$ and S is a set of Services, a user is allowed to access $\{S_1, S_2, S_3..\}$. After accessing the service, the user is requested to fill in the Feedback form which is in the form of a questionnaire and the values are taken on a scale of 0-1 for trustworthiness and reputation scores. The Service Provider is also requested to fill up the pending feedback form. Thereby, making the transaction complete. An Intrusion Detection System and Audit Trails checks the information provided to be from the intended players. The trust values and recommendation values are updated in their respective databases. If the values are reported from the malicious origin, the values are discarded and the malicious domains are blacklisted, by imposing a penalty. TUX-DB is responsible for maintaining the database.

The functioning of components has been elaborated in the next section therefore not explicitly mentioned here in Architecture.

4.2.5.1 Identity Management System

An identity management system maintains repository of the user credentials and interacts with various Trusted Third Parties (TTP) to check for validation of the credentials provided by the user. The Identity Management System incorporates authentication, authorization and trust related functions. IDS is comprised of authorization actions. An authorization action is described by $a\{s_1, s_2, t, r_1..., r_2..\}$ where a is the action name, s_1 is the authorization operation executor, s_2 is the receptor of privilege, t is the specific operation relative to the privilege. The 3-tuples $\{s_1, s_2, t\}$ expresses that s_1 authorizes p that relative to t to s_2 . Here, $r_1....r_i$ are the environmental or system status. In proof

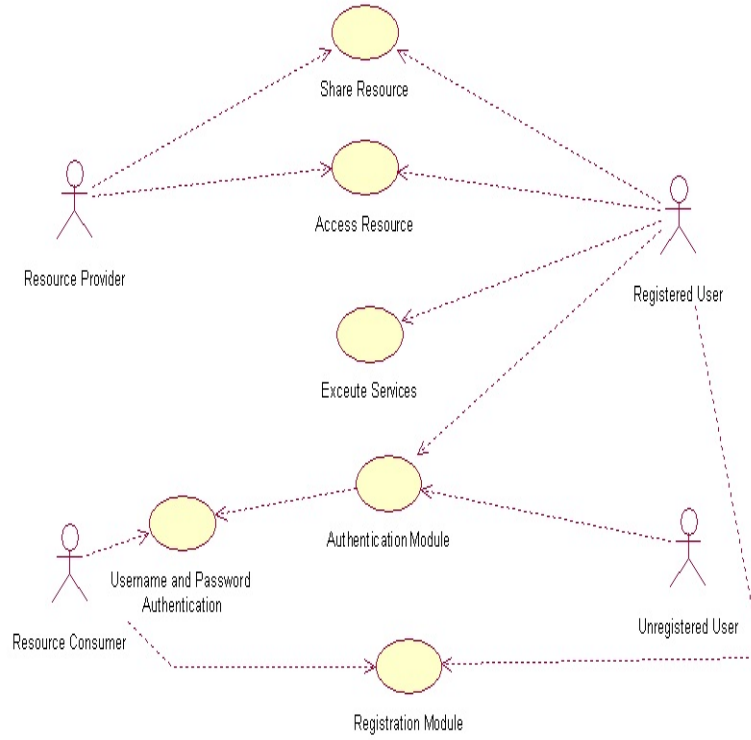


Figure 4.3: Use Case Diagram for User Authentication in TUX-TMS

of compliance, Service Provider/Resource Owner can make access decision on certain resource to Service Requester/Resource Consumer. So, Service Provider can validate Service Requester's attribute whether satisfies local policy or not firstly (attributes coded in certificates, which were issued by certified authorities) and then grant or deny the access request. This may be expressed as Equation 4.2.

$$A \xrightarrow{\text{credentials}} B \Rightarrow B \xrightarrow{\text{credentials}} A \Rightarrow A \text{ validates credentials} \Rightarrow B \text{ permission } p \quad (4.2)$$

Similarly, the Service Provider can make access decision on certain resource in reputation check. Differently, Service Provider should query historic observation on Service Requester credentials whether they satisfy local policy firstly (direct interoperation record, failed affair induce passive opinion and successful affair induce positive opin-

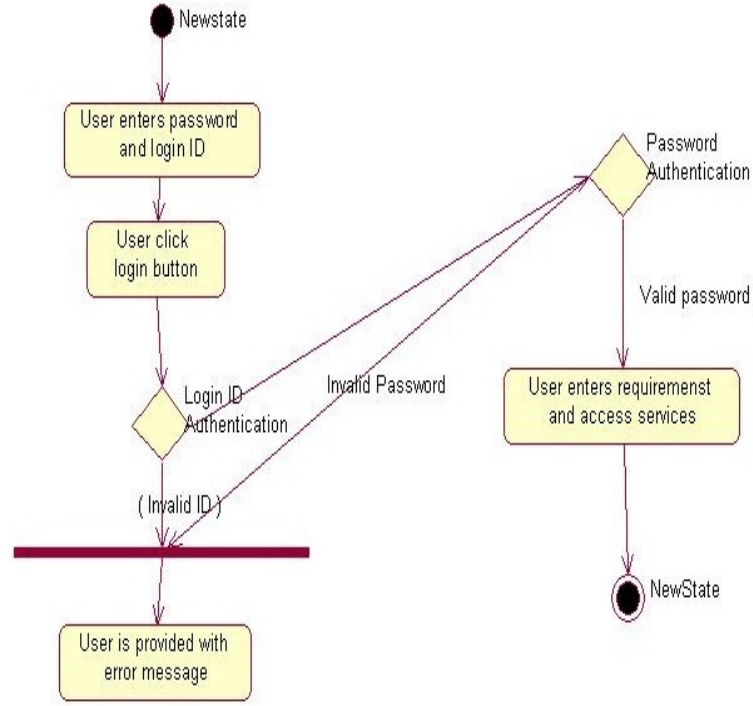


Figure 4.4: Activity Diagram for User Authentication in TUX-TMS

ion) and then grant or deny access request based on the records available. This may be expressed as Equation 4.3.

$$A \text{ check reputation about } B \Rightarrow B \text{ satisfies local policies} \Rightarrow B \text{ has permission } p \quad (4.3)$$

To trust relationship based on proof of compliance, it is inherently transitive. Suppose A may not be the owner of p but possess the privilege to delegate p to others. A delegate p to B if B's attributes satisfies local policies, at the same time announces local policy to B. Furthermore, if B trusts C and then B delegate p to C if C's attributes satisfies B's policy, which must ensure more strict constrain than A's policy, well than

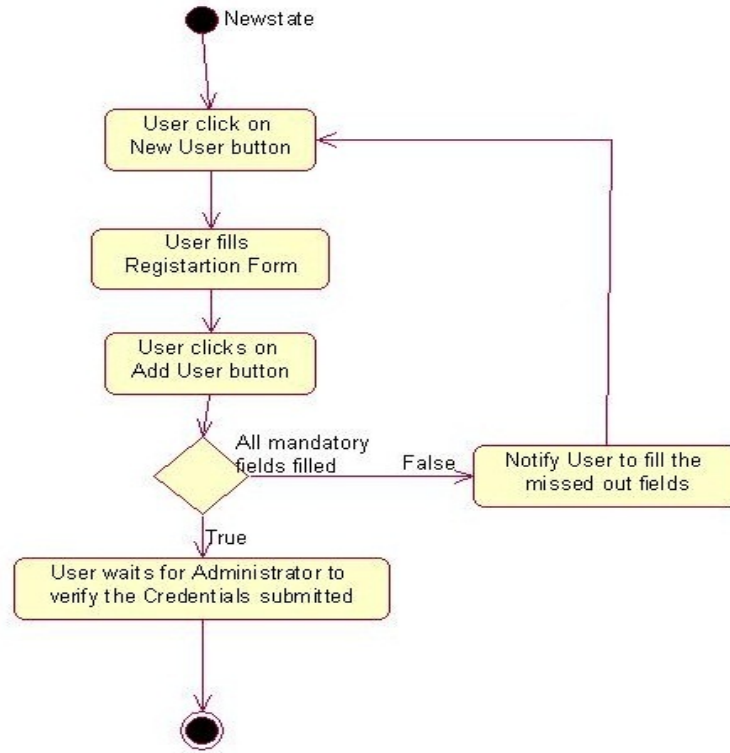


Figure 4.5: Activity Diagram for New User Registration in TUX-TMS

A trust C has p . This may be expressed as Equation 4.4.

$$A \xrightarrow{\text{delegate } p} B, A \xrightarrow{\text{delegate } p} B \Rightarrow C \xrightarrow{\text{credentials}} B, B \text{ validates credentials} \quad (4.4)$$

To trust relation based on reputation check, it can also be transitive. Each affair can acquire feedbacks, which can be collected, aggregated and distributed among domains due to that a domain's past history becomes informative to predict its future behavior. Feedback among the domains can be classified under trustworthiness and recommendation feedback. Trustworthiness denotes the history of a transaction with the other domains. However, recommendation feedback presents that a domain trusts the ability of another domain to recommend it to others. Whereas, unrestricted calculation of transitive trust value may be not accurate. Suppose A trust B with reputation value

c_1 , B trust C with reputation value c_2 . Due to transitive nature of trust, A should trust C with reputation value c_3 (depends on the application circumstance). It is determined in Grid environment in terms of context factor of the reputation or recommendation values of a domain.

The proof of compliance is based on credentials, so each domains must hold credential database. However for TUX-TMS, a central repository for storing credentials is considered. Otherwise, each domains will be required to hold credentials of the domains interacting with it.

4.2.5.2 Trustworthiness Base

The trustworthiness is evaluated and calculated from past transaction scores and the feedback provided for the service provided by both Service Requestor and Service Provider. The feedback is taken from the Service Provider/Service Requester in form of a questionnaire which has value which lies between 0-1. The trustworthiness score is calculated by aggregating the values of entities for a particular domain. The score are added for accumulated values which are normalize by number of transactions.

Add_Feedback(ETT)

Input:*ETT* **Output:***DET, ETT*

for $i = 1$ *to* $length(DET)$ *do*

receiver $\leftarrow SR$ *of* $DET(i)$

if *receiver* = R *then*

$TEL\_SR \leftarrow SR$ *of* TEL *of* $ETT(i)$

endif

provider $\leftarrow SP$ *of* $DET(i)$

if *provider* = P *then*

```

TEL_SP <= SP of TEL of ETT(i)
endif
endfor

ETT_SR <= TEL_SR + sr
ETT_SP <= TEL_SP + sp

for j = 1 to length(DET) do
  if R = Domain Name of Domain Evaluation Table then
     $\sum F_s$  of Domain Evaluation Table <=  $\sum F_s$  of Domain Evaluation Table + sr
     $\sum R_c$  of Domain Evaluation Table <=  $\sum R_c$  of Domain Evaluation Table + sr
     $\sum N_t$  of Domain Evaluation Table <=  $\sum N_t$  of Domain Evaluation Table + 1
  endif

  if P = Domain Name of Domain Evaluation Table then
     $\sum F_s$  of Domain Evaluation Table <=  $\sum F_s$  of Domain Evaluation Table + sp
     $\sum R_c$  of Domain Evaluation Table <=  $\sum R_c$  of Domain Evaluation Table + sp
     $\sum N_t$  of Domain Evaluation Table <=  $\sum N_t$  of Domain Evaluation Table + 1
  endif
endfor

```

The Add_Feedback() algorithm updates trustworthiness and reputation values in Entity Transaction table (ETT) of the transactions committed between Service Requester and Service Provider. The values are aggregated and updated in Domain Evaluation Table (DET). In Add_Feedback(), F_s stands for feedback scores, R_c is used for recommendation value, N_t stands for number of transactions and TEL stands for Total Entity Value. SR and SP stands for Service Requester and Service Provider, respectively.

4.2.5.3 Reputation Base

The reputation engine provides the recommended score of an entity or domain. The reputation engine provides the aggregated scores of the recommendation provided for the entities and domains. The recommended values are taken in form of a questionnaire submitted by the Service Requester/Service Provider. The recommendation scores further are stored and normalized by the number of domains submitting the scores. This normalization takes care of a malicious domains trying to increase reputation values of a malicious domain.

4.2.5.4 Trust Inheritance Module

The entity which is a part of a domain inherits some value from the domains depending upon the credibility of the domain. For a malicious domain, the value is very low. The values are inherited using Equation 4.7 where α , β and γ stands for belief, disbelief and uncertainty. The higher the trustworthiness and reputation of domains, the higher is the value inherited. This inherited value allows domains and entities to access more services in a Grid.

Trust_Inheritance(DET, domain)

Input:*DET* **Output:***DET*

for $i = 1$ *to* $length(DET)$ *do*

if $DET[i].Domains = domain$ *then*

$DET[i].safety\_factor \leq 100 - DET[i].risk$

$DET[i].TrustInheritance \leq DET[i].TEL * DET[i].safety\_factor$

endif

endfor

The *Trust_Inheritance()* algorithm allows an entity to inherit value from the domain,

it is part of. The values if calculated using `safety_factor` as mentioned in Equation 4.7 which is the risk involved in transaction with a domain.

4.2.5.5 Risk Assessment Generator

The risk involved in transacting with a domain is assessed taking the rank of the domain in the Domain Evaluation Table (DET). DET tables stores the values on the basis of the parameters as discussed in the previous section. Higher the rank, minimal is the risk involved in transacting with the domain. The risk is categorized using Table 4.3. The domain chart is divided into quartiles and the domains are rated as per Table 4.3. The domains lying in catastrophic and critical risk status are considered very risky to transact with. However, the domains having moderate and negligible risk are considered safe to transact with.

Risk_Assessment(DET, domain)

Input:DET **Output:**DET

```

for i = 1 to(DET) do
  if DET[i].Domains = domain then
    DET[i].risk <= rank/max_rank * 100
  endif
endfor

```

`Risk_Assessment()` calculates the risk involved in interacting with one particular domain and is calculated by dividing the rank of a domain with total domains in the DET. A domain which is viewed as being ‘not so risky’ is normally assigned a higher level of trust, while a highly trusted domain is considered to be of low risk. The risk is assessed by percentage of rank of a domain by total number of domains in Domain Evaluation Table (DET). The risk is calculated using Equations 4.8 and 4.9.

4.2.5.6 Trustworthiness and Reputation Decay Module

The trustworthiness and reputation values decay on the basis of the inactivity of a particular domain and triggered after a period of time. The lowest transaction considered are five of low trust context factor. The trust decay factor decays the trust and reputation value and they are to be regained again by the domain for future use purposes. The values are decayed using Equations 4.5 and 4.6.

Decay_Trust(DET)

Input:*DET* **Output:***DET*

for i = 1 to length(DET)do

$$\sum T_u \text{ of } DET(i) \leq \sum T_w \text{ of } DET(i) - T_{decay} \text{ of } DET(i)$$

$$\sum R_c \text{ of } DET(i) \leq \sum R_c \text{ of } DET(i) - R_{decay} \text{ of } DET(i)$$

$$T_{decay} \text{ of } DET(i) \leq \sum F_s \text{ of } DET(i) / \sum N_t \text{ of } DET(i) * 100$$

$$R_{decay} \text{ of } DET(i) \leq \sum R_c \text{ of } DET(i) / \sum D_n \text{ of } DET(i) * 100$$

endfor

Here, Decay_Trust()

decays the values of trustworthiness by total trustworthiness value by total number of transactions incurred and reputation values by number of domains transacted with.

Here, D_n stands for total number of domains with whom transaction has taken place.

4.2.5.7 Trust Inference Engine

A trust inference engine takes the value of decayed Trustworthiness and Reputation scores, Risk Assessment and Trust Inheritance as per the trust context factor for calculating the threshold value to allow a domain to transact.

Trust_Inference_Engine(DET, domain, r, sr, rp, R, P)

Input:*DET, domain, r, sr, rp, R, P* **Output:***DET, TST*


```

callDecay_Trust(DET)
callRisk_Assesment(DET, domain)
callTrust_Inheritance(DET, domain)
threshold_SR <= w1 * TdecayofDET(r) + w2 * Rdecay of(r) + w3 * Trust_InheritanceofDET(r) - w4 * Risk_Assesment of DET(r)
for i = 1 to length(DET)do
if threshold of(i) < threshold_SR then
TST_ID <= i
TST_SR <= threshold_SR
endif
endfor

```

The trustworthiness of the domain with high feedback scores, increases. The threshold value is used to determine a demarcation between domains with higher reputation. The Trust_Inference.Engine() algorithm takes the value of risk assessment, trust inheritance and trust and reputation decay values before determining the threshold. The threshold is further required for providing services to the Service Requesters as per their values. Here, weights w1, w2, w3 and w4 are assigned to the risk assessment, trust inheritance and trust and reputation decay values on the basis of trust context factor. All other functions are called here to calculate threshold such as Decay_Trust, Risk_Assesment and Trust_Inheritance.

4.2.5.8 Scoring systems

Reputation systems depend on scores to provide some meaning to the ratings as a whole. The scores can be very simple or involve multiple scales and complicated calculations. TUX-TMS scoring system possesses the following qualities which are also

deemed essential for a good scoring system.

- Accurate for long-term performance.
- The system reflects the confidence (the likelihood of accuracy) of a given score. It can also distinguish between a new entity of unknown quality and an entity with bad long-term performance.
- Weighing current behavior: The system recognizes and reflects recent trends in entity performance. For instance, an entity that has behaved well for a long time but suddenly goes downhill is quickly recognized and no longer trusted.
- Efficient: It is convenient if the system can recalculate a score quickly. Calculations that can be performed incrementally are important.
- Robust against attacks: The system should resist attempts of any entity or entities to influence scores other than by being more honest or having higher quality.
- Amenable to statistical evaluation: It should be easy to find outliers and other factors that can make the system rate scores differently.
- Private: No one should be able to learn how a given rater rated an entity except the rater himself.
- Smooth: Adding any single rating or small number of ratings doesn't jar the score much.
- Understandable: It should be easy to explain to people who use these scores what they mean - not only so they know how the system works, but so they can evaluate for themselves what each score implies.
- Verifiable: A score under dispute can be supported with data.

The ratings have been weighed by the credibility of each rater. That is, certain people contribute more to the score of a given entity based on their past predictive ability. The feature of global credibility for each entity provides a great deal of flexibility and thus stronger predictive value.

4.2.5.9 IDS and Audit trail

An Intrusion Detection System (IDS) is a device or software application that monitors network and/or system activities for malicious activities or policy violations and produces reports to an Administrator. While the default view of the audit trail is to see all entries made in the last 10 days, the entries can be filtered and grouped to show detailed information on changes made a certain way or over a certain time. In addition, filters can be stacked on top of each other, as well as on top of grouping levels. Grouping for the audit trail entries works just as it does for the item views. A series of grouping options are offered based on available fields. When a group by field is chosen, the view of the audit entries shifts to group entries by the values of that field. Available fields for grouping are: Item Id, Item Type, Project Name, Audit Type, Audit Source, Audit Date, User Name. In addition to grouping data by fields, the audit trail entries can be filtered to show only specific entries. These filters include: Item Type, Audit Type, Audit Source, User and Date Range. User filters by the name of the user who made the change that cause the audit entry.

4.2.5.10 Bootstrapping

One of the major step for a reputation-based trust system is how to get started. If users make choices based on the scores that are available to them, but the system has not yet

collected enough data to provide useful scores, what incentive do buyers have to use the system? More importantly, what incentive do they have to contribute ratings to the system? Bootstrapping is a much more pronounced issue in a centralized system than in a decentralized system. This is because in a decentralized system, each user develops his own picture of the universe: he builds his trust of each entity based on his own evidence of past performance and on referrals from other trusted parties. Thus, every new user effectively joins the system “at the beginning,” and the process of building a profile for new users is an ongoing process throughout the entire lifetime of the system. In a centralized environment, on the other hand, ratings are accumulated across many different transactions and over long periods of time. New users trust the centralized repository to provide information about times and transactions that happened before the user joined the system. In a newly developed system or for a new entity in the system, the choice of the default reputation score is critical. If it’s easy to create a new identity and new users start out with an average reputation, users who develop a bad reputation are encouraged to simply drop their old identities and start over with new ones. One way to deal with this problem is to start all new users with the lowest possible reputation score; even users with a bad track record will then have an incentive to keep their current identities. Another approach to solving this problem is to make it difficult to create a new identity. For instance, this can be done by requiring some proof of identity or a monetary fee for registration. Tying the user to her real-world identity is the simplest, and probably the most effective, way to reduce abuse - but only if it’s appropriate for that system. This approach is considered by TUX-TMS. The next section focusses on implementation details of TUX-TMS.

4.2.6 Implementation details of TUX-TMS

TUX-TMS has been implemented as a Reputation Based Trust Management System. A heterogenous computing environment has been created using Pentium IV machines running Windows and Linux. The “TUX-TMS” System has been executed using the .Net framework to implement TUX-TMS components and MS-SQL Server has been chosen to store the information in a database which can be further divisioned onto other database repositories in future. The information has been stored in form of tables and procedures & functions have been implemented to query, provide and update information. The resources considered are computational, data and printing.

TUX-TMS allows Service Requesters and Service Providers to assess trustworthiness of each other as well as evaluate and establish trust relationship among them. An activity diagram in Figure 4.6 depicts the information flow in TUX_TMS(). The activity diagram depicts that an end user can only request a service by authenticating into TUX-TMS. If the end user is registered, he is authenticated and allowed access, else requested to register. The values of identity trust is provided by Identity Management System for submitting secure credentials, reputation and trustworthiness base provides their values and likewise values are provided for risk assessment and trust inheritance of an entity joining afresh. Trust Inference Engine takes all the values and calculates trust threshold according to a particular context and allows services to the Service Requester. Here, Service Requester is also allowed to calculate risk involved in assessing services of the Service Provider. After, selecting Service Provider and accessing Service, Service Requester is requested to input feedback values for Service Provider. For a successful transaction, Service Provider is also required to submit feedback values for the Service Requester. The values are checked by IDS and Audit trails to be coming from reliable source and trustworthiness and reputation bases for Service Re-

quester and Service Provider is updated likewise. The same procedure is conducted by TUX-TMS() algorithm mentioned below:

TUX-TMS(ETT, DET, TST, SR_ID, SPD, domain)

Input: *ETT, DET, TST, SR_ID, domain* **Output:** *ETT, DET, TST*

If authenticated then

If authorized then

Resource_request

T_w <= Trust_Inference_Engine(DET, domain, r, sr, rp, R, P)

Resource_requested <= select k from TST_ID

If trust_requirement of ID = Resource_requested from SPD < T_w then

Notify_ID about request

If ID approves then

T_{id} <= Transaction

Transaction occurs

Transactionends

Endif

Endif

If enter_feedback then

Valid_tid <= Verify_Transaction(t_{id})

If Valid_tid = true then

Valid_user <= Perform_IDS(SR_ID)

If Valid_user = true then

Add_feedback(TEL_sr, TEL_sp, R, P)

Endif

Endif

Endif

Endif

Else

Register

Endif

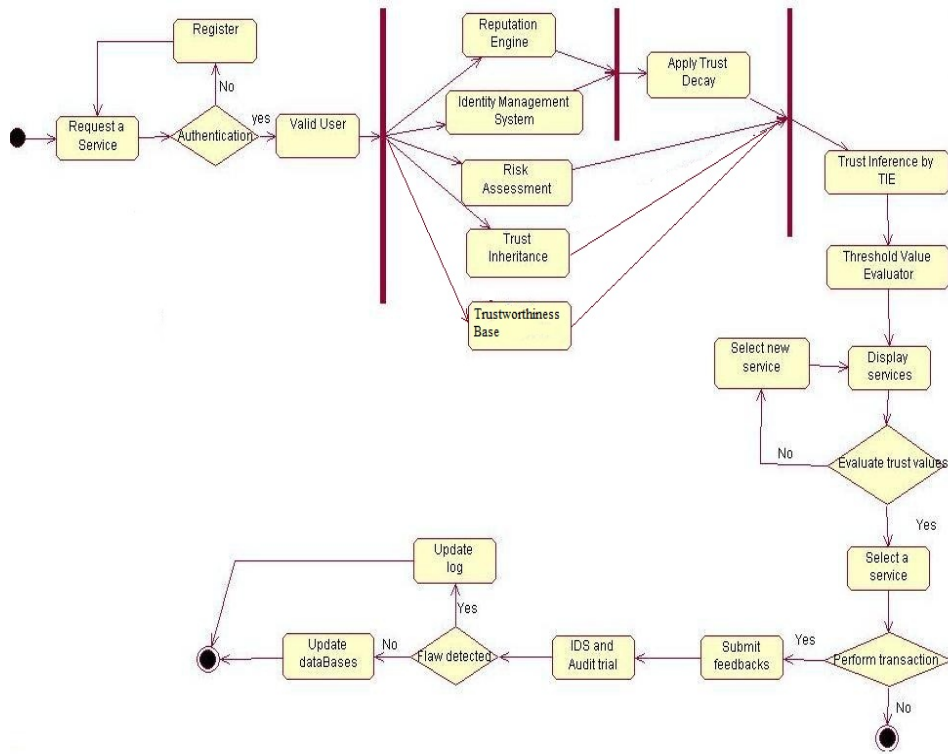


Figure 4.6: An activity diagram of TUX-TMS

TUX-TMS() allows Services to Service Requester by calculating trust threshold T_w . The steps shown in the algorithm are implemented as modules and explained in forthcoming sections. The steps start by authenticating an end user into the system.

4.2.6.1 Authentication

Firstly, the Identity Management Component comes into play. It uses the credential information from an entity requesting services. There are three types of users in TUX-TMS. They are Service Requester as shown in Figure 4.7, Service Provider as shown in Figure 4.8 and Administrator as shown in Figure 4.9 and 4.10. TUX-TMS uses Role Based Access Control (RBAC) for authorization and access control purposes. Here, the users have different access permission based on their roles. The credentials has been designed using .Net tools. If the entity is using X.509 certificates, the certificates are issued by the certificate authorities. In TUX-TMS, administrator is chosen as certification authority. Certificates bind the entity's Distinguished Name (DN) to its public and private key. Public key is written with in the certificate whereas private key is with the entity. An entity needs to present certificate and showcase the possession of private key in order to prove their identity to system. The administrator is responsible for authorizing users to use resources by verifying them. In case of multiple credentials (such as Kerberos tickets, X.509 certificates and username/password pair) for multiple services, an entity's needs to maintain multiple credentials which can be stored using any Credential Management Service (CMS) and the credentials can be requested accordingly for future purposes. On the basis of trust threshold users are allowed to use the resources.

4.2.6.2 Authorization

Authorization grants specific type of access to specific requesters based on their authentication, what services/resources they are accessing, current state of the system and their conformance to established authentication, trust and other security policies. TUX-TMS grants specific type of access to specific requesters based on their authen-

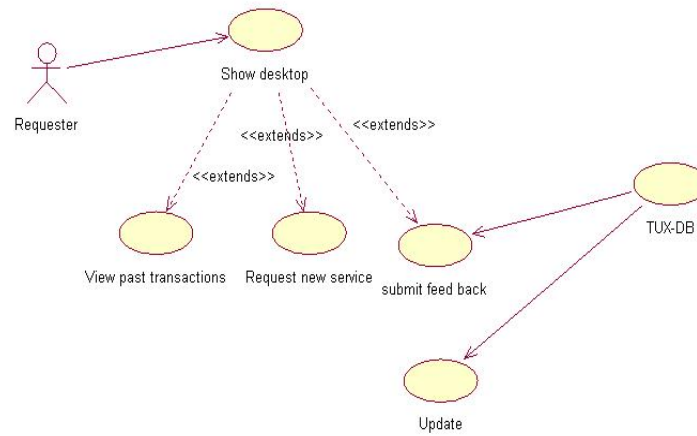


Figure 4.7: Use Case of Service Requester in TUX-TMS

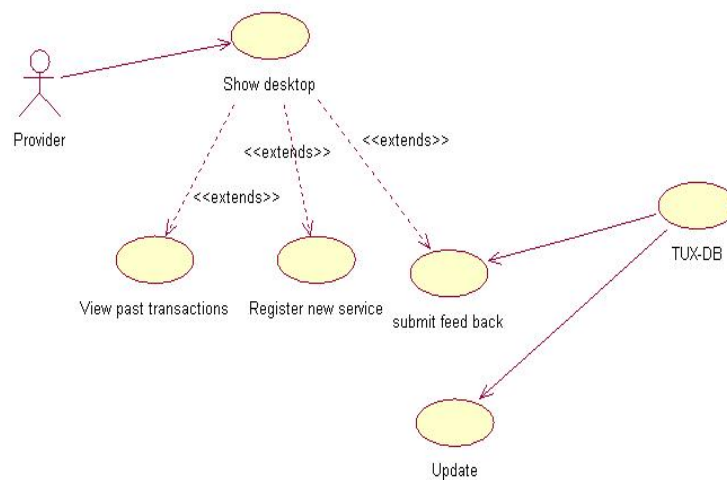


Figure 4.8: Use case of Service Provider in TUX-TMS

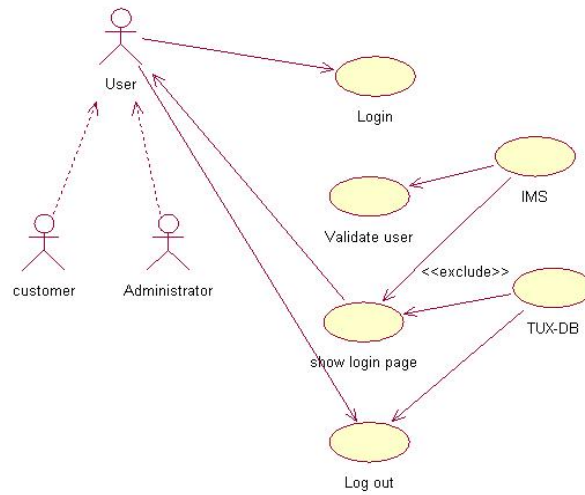


Figure 4.9: Use case of Administrator in TUX-TMS

tication, identity trust, behavior trust and their conformance to established trust. In Grids, the number of entities requesting services or the resources are very large. They have different access rights and control depending on the domains, they are part of. It becomes tough for Service Providers to cater to a large number of users and maintain access information for all. TUX-TMS provides access to domains which the entities are part of. Access control policies are pivotal for TUX-TMS. Policies define who have specific characters (certain attributes coded in certificates or reputation values derived from historical accumulation of past behavior or recommendation values) can access certain resources. A finite set P , of possible policies, each of which is comprised of the following two forms:

1. A logical proposition is comprised of a combination of and, or and attributes in T . An attribute in a policy will be true with respect to a principal k if k has that attribute.

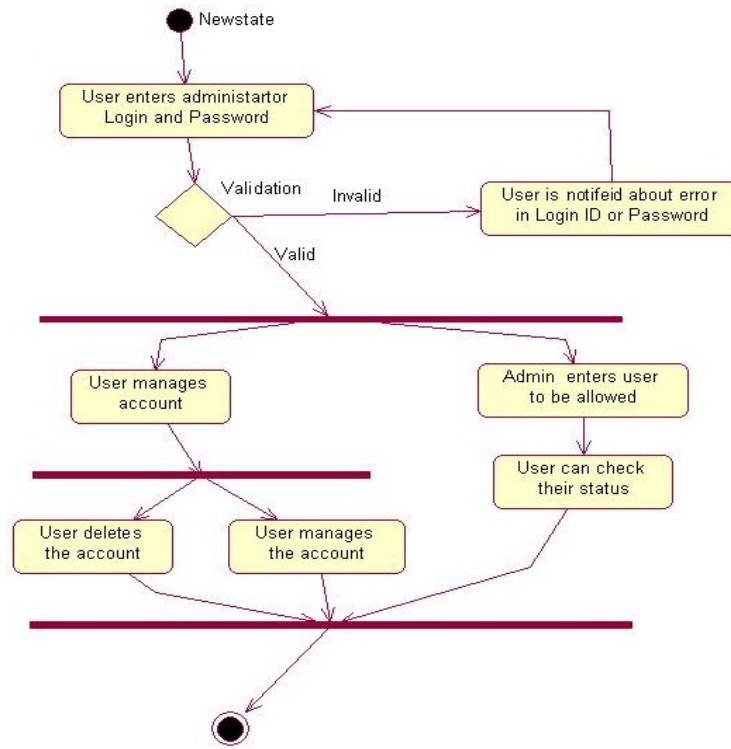


Figure 4.10: Activity Diagram for Administrator in TUX-TMS

2. A mathematical expression about one or multi numerical values and variables, which may be one dimension or multi-dimensions.

The source domains are required to maintain information about access control of the entities to make the system scalable. The entities inherit scores of their domains. For brevity purposes, domains are considered to access resources. The authorization is implemented in TUX-TMS using filter-in and filter-out approach. Through filter-out component, an entity is provided access rights to make use of a service. Through, filter-in component an entity can enter the domain to access services based on the credentials submitted. This makes the authorization more scalable.

4.2.6.3 Scoring System

One of the most interesting aspects of the Reputation Server is the scoring system, the manner in which it computes reputations from all of the feedback that it has gathered. First, the feedback chain may be sensitive information, because it includes a description of previous transactions. Scoring algorithms can mask details and protect the privacy of previous raters. This trade-off between hiding and revealing data is more subtle than encryption. Encryption seeks to transform data so that, to the unauthorized reader, it looks as much like noise as possible. With reputation, there is a need to simultaneously mask private aspects of the transaction history - even to the authorized reader - while allowing some portion of the history through so it can influence the reputation. Some of this is accomplished simply by compressing the multiple dimensionality of the history into a single point, perhaps discretizing or adding another noise source to the point to constrain its dimensionality. Furthermore, the Reputation Server has a more global view of the feedback data set than one can learn from viewing a simple history listing, and it can include other sources of information to give a better answer about reputation. The scores are taken on likert scale and stored and evaluated for future purposes.

4.2.6.4 Computing trust values

TUX-TMS calculates the trustworthiness and reputation values of the entities and domains by using the following components: In computing trustworthiness and reputation, several issues have to be considered. First, trust decays with time. For example, x trust y at level l based on past experience five years ago, the trust level (Table 4.1) today will be likely very low unless they have interacted since then. The trustworthiness and Reputation values stored in TUX-DB are called and decay function is applied when the values are used. The decay functions are mentioned using Equation 4.5 and

4.6. In Equation 4.5, T_{Decay} is the trustworthiness decay calculated by dividing trustworthiness value (T_u) by total number of transactions (N_t). In Equation 4.6 (R_{Decay}) is the reputation decay calculated by dividing (R_c) by total number of domains (D_n), that have interacted.

$$T_{Decay} = T_u / N_t \quad (4.5)$$

$$R_{Decay} = R_c / D_n \quad (4.6)$$

This gives a value of an entity. Then, if it inherits some trust value using Equation 4.7 from the domain, it is part of is checked and value is inherited by T_{fin} using α stands for belief, β stands for disbelief and γ stands for plausibility. For the entities that are new, inherit only trustworthiness values T_u of the domains, they are part of. However, the entities who are entering after a period of time inherits reputation values of the domains they are part of. Finally, if the entity is transacting, no value is assigned, where plausibility is kept high and entity is allowed to gain values for actual transactions.

$$T_{fin} = \alpha * T_u + \beta * R_c + \gamma \quad (4.7)$$

Then, the risk assessment component shows the risk involved in interacting with entity is calculated using Equation 4.8 and 4.9. The domains are ranked in Domain Evaluation Table (DET) according to their trustworthiness and reputation values. The highest rank in DET shows domains with high scores of trustworthiness and reputation values. In Equation 4.8, the domains are divided into quartiles (as shown in Table 4.3) and the domains lying in catastrophic and critical arena are not allowed to access sensitive Services. In Equation 4.9 the values are inherited by subtracting T_{fin} from

Risk involved in accessing Services calculated using Equation 4.8.

$$R_a = R_{ank}/T_{dn} \quad (4.8)$$

$$T_i = T_{fin} - R_a * 100 \quad (4.9)$$

The Risk assessment Component is responsible for identifying the risks. After identification, the risks are analyzed. In this step, the risks are quantified. This means assigning probabilities to the risks and estimating the impact to the Service Provider of these risks occurring. Once this is done, overall risk values are calculated. The above step corresponds to a traditional quantitative risk model. However, due to the imprecise nature of the data, qualitative labels are normally used for both the probabilities and impact estimates. The risk probabilities and impacts are shown in the Table 4.3. The risk probabilities may be: low, medium, high or very high and the impacts of risk may be: negligible, marginal, critical or catastrophic. The calculation of the overall risk value is done by evaluating the impact/probability matrix, which is constructed from the assignments. After analysis, a list of highest risks is identified. Then, a plan is formulated to address each risk. In planning, the following things are covered: Why is the risk important? What is needed to track the risk? Who is responsible for the risk management activity? What resources are needed to perform the activity? A detailed plan of how the risk will be prevented and or corrected needs to be formulated. This includes an action plan - to resolve an immediate risk and a contingency plan, to monitor the risk and trigger a predetermined response. This way a risk can be tracked. This is kept into consideration for future extensibility purposes. The number of users are divided using quartiles and likewise using the Table 4.3 only the users with

Table 4.3: Risk Probabilities and Impacts of Risk

Risk Probabilities	Impacts of Risk
Very High	Catastrophic
High	Critical
Medium	Marginal
Low	Negligible

whom the risk is below catastrophic and critical level are allowed to use the services. This threshold is applied so as to avoid incurring heavy security losses and threats to system. Likewise, the trust information is available to the Service Requester to the Service Providers. As the risk considered here are in terms of security threats, this can be further extended to include risk in terms of financial losses.

Using the above equations and data structures trust of a domain can be calculated which is presented to the requesting domain.

4.2.6.5 Querying trust information

The user interface - that is, the way of presenting scores and asking questions - is a crucial element of a reputation system. Scores cannot simply be static universal values representing the overall quality of an individual. Since a score is an attempt to predict future performance, each score must be a prediction for a particular context. That is, the user interface must allow participants to query the system for the likelihood of a successful transaction for their particular situation. The more flexibility a client has, the more powerful and useful the system is (so long as users can still understand how to use it). The user interface must also display a confidence value for each score - that is, how likely the score is to reflect the reality of the subject's behavior. The Service Requester may use this information to use the services and can calculate the risk involved in using the services of the domains and can use services from trustworthy

Service Providers. The information is queried using SQL queries and information has been presented using Views in SQL. Figures 4.11 and 4.12 shows Stored Procedures and Functions in TUX-DB to query and update values.

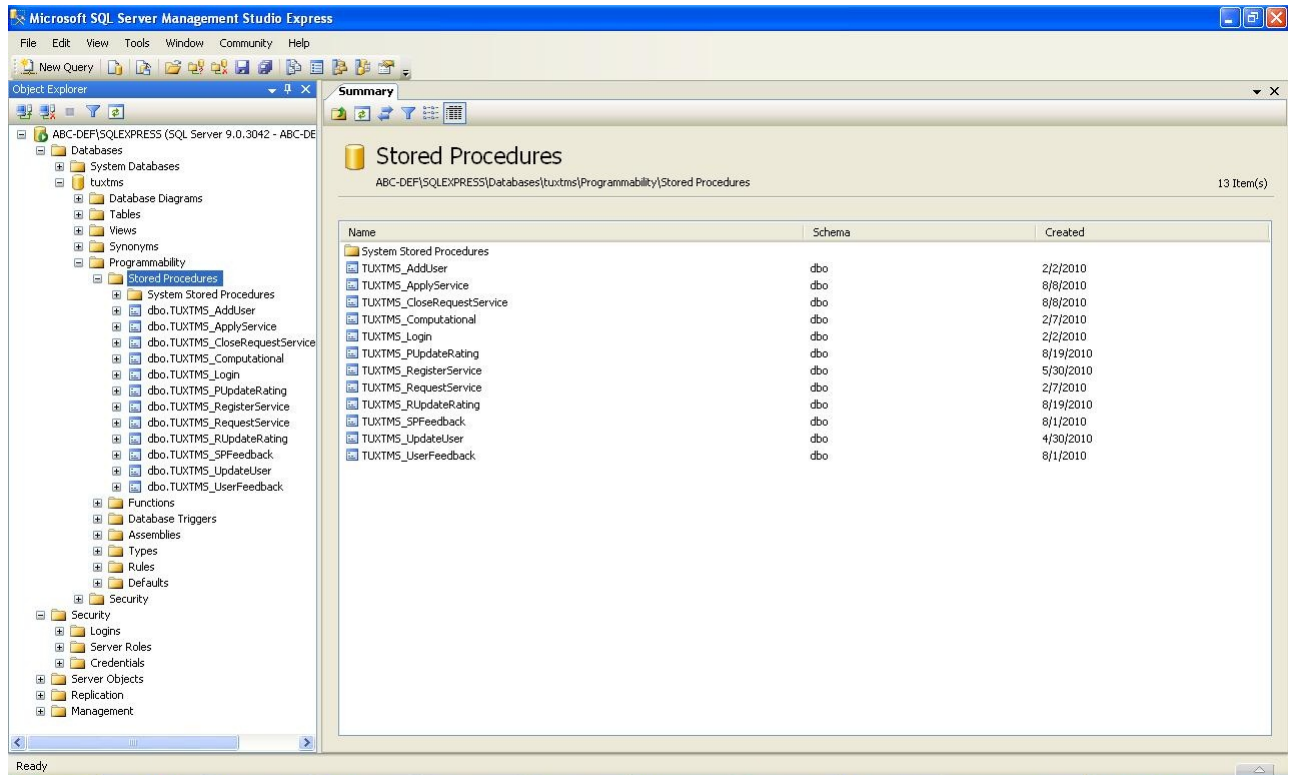


Figure 4.11: Stored Procedures in TUX-DB to query and update values

4.2.6.6 Storing trust values

After the Service Requester has used the service, he is requested to provide feedback scores for the Service Provider in form of a questionnaire. TUX database (TUX-DB) stores the feedback information of the Service Requester as well the Service Provider. This TUX-DB is also used to obtain interaction histories concerning the Service Requesters and Service Providers and their credentials. Context data is stored so reputation can be calculated at varying granularity. Table 4.4 shows detailed information of tables for implementation purposes. Figure 4.13 depicts the tables used to store values

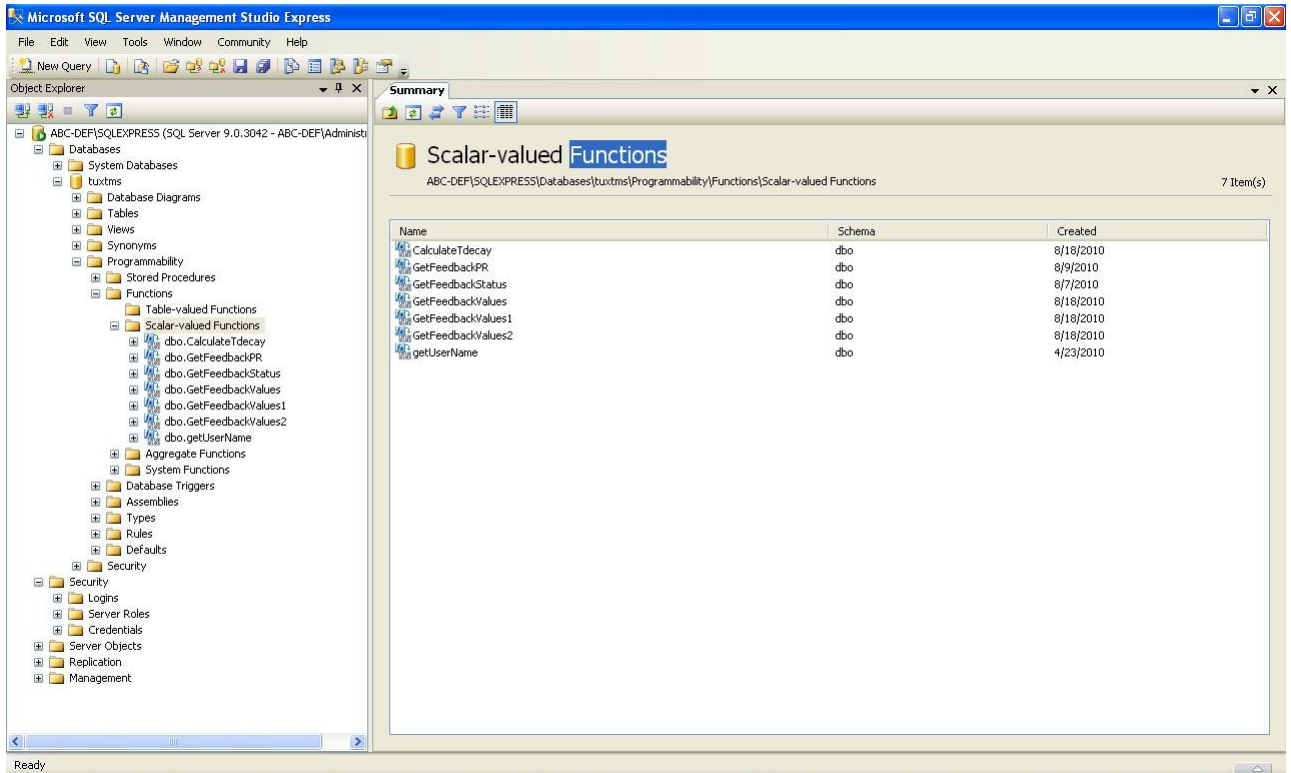


Figure 4.12: Functions in TUX-DB to query and update values

in TUX-DB.

4.2.6.7 Intrusion Detection and Audit Trails

OSSEC [230] was used for IDS purposes which is a free, open source host-based intrusion detection system (IDS). It performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, time-based alerting and active response. It provides intrusion detection for most operating systems, including Linux and Windows. It has a centralized, cross-platform architecture allowing multiple systems to be easily monitored and managed. Intrusion Detection and Audit Trails are conducted periodically and the entities and domains giving malicious information are blacklisted and penalized. It checks if the information comes from the genuine source else the information values are discarded and not updated in the TUX-DB.

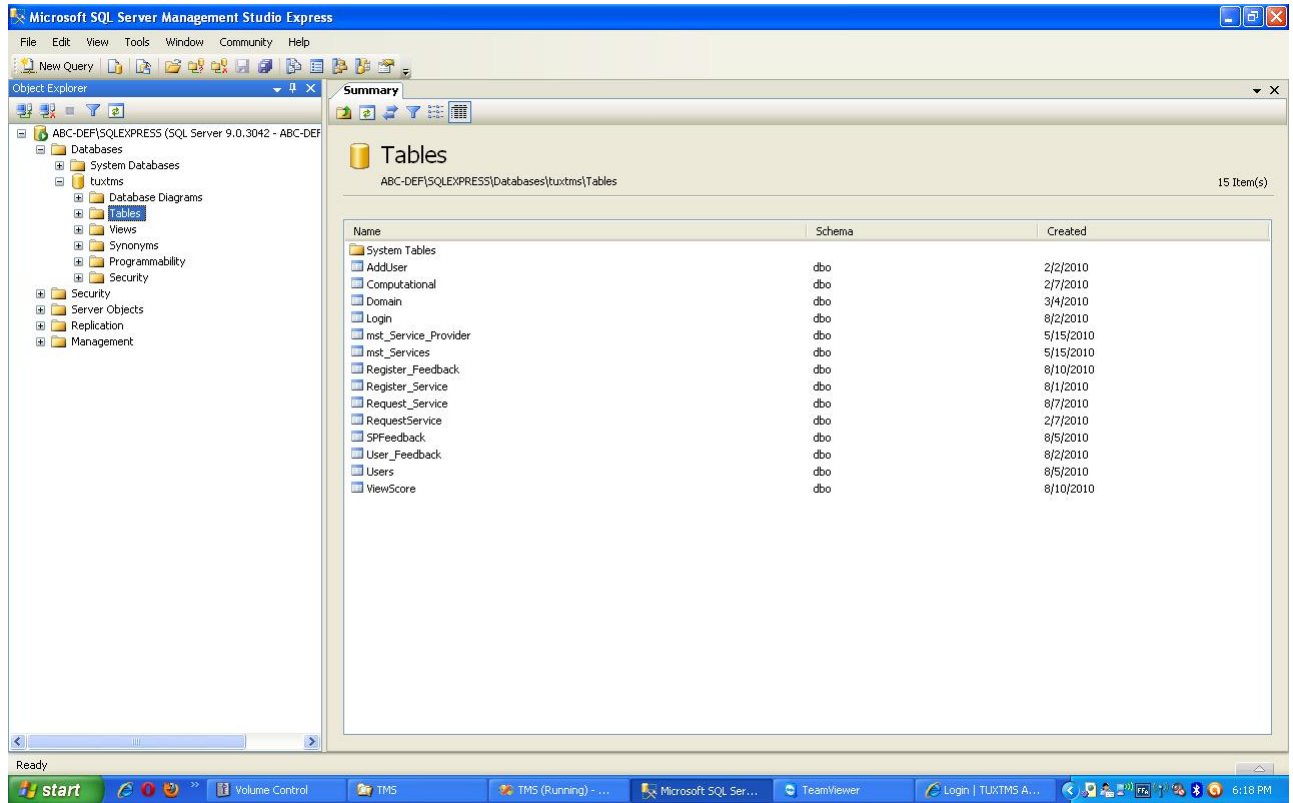


Figure 4.13: Tables in TUX-DB to store values

TUX-TMS is a flexible Reputation based system which considers positive aspects of both policy based and reputation based approaches for designing trust systems. It caters to all the basic criteria for judging the quality and soundness of reputation computation engines as proposed by Dingledine et. al. [191]. TUX-TMS caters relevant parameters deemed essential to secure Grid environments. Table 4.5 compares TUX-TMS with existing trust management solutions.

4.3 Conclusions

The chapter focussed on TUX-TMS, the proposed trust management solution. It described taxonomy and parameters of TUX-TMS. The architecture of TUX-TMS which has various components such as Identity Management System, Trust Inference Engine,

Table 4.4: Database Tables

Tables	Information contained
Users	Stores the information related to the User in a centralized repository
Register Services	Stores the information regarding the Services registered
Service Provider Feedback	Stores the information regarding feedback submitted by a Service Provider for Service Requester
Service Requester Feedback	Stores the information regarding feedback submitted by a Service Requester for Service
View Scores	Views the scores of various Service Provider

Risk assessment, IDS and Audit Trail, Trust and Reputation Decay, Trustworthiness, Reputation Collection, Feedback and Trust Inheritance have also been covered. The functioning of the components has been discussed in detail here. In the discussion of TUX-TMS, it was shown that it fulfills the requirements of the reputation based trust management for Grid environments, which are dynamic and heterogeneous in nature. The implementation details were presented at the end of the chapter.

Table 4.5: Comparing existing Trust Management solutions with Proposed solution

Types of Solutions	Features	Existing Solutions	Proposed Solution
Policy based Trust Management	Most policy based trust management systems focus in protocols for establishing trust in a particular context. Some make use of a trust policy language to allow the trustor to specify the criteria for a trustee to be considered trustworthy	Public Key Certificate[146], PGP trust model[146], X.509 trust model[147] [148], PICS[151], PolicyMaker[152], KeyNote[152], REF-EREE[153], SD3[154], Fidelis[155], TrustBuilder[156] [157], TCPA[158], Poblano[159], RT[160], Active Trust Management System[161], Kagal[162]	Policy is used in proposed model for for identity purposes
Reputation based Trust Management	Reputation based trust management system focusses on trust computation models capable of estimating the degree of trust that can be invested in a certain party based on the history of its past behavior	Pretty Good Privacy (PGP)[164], P-Grid[165], Eigen-trust[166], Abrams[167], XenoTrust[168][169], PeerTrust[170], SeGO [171][172], GridEigen-Trust[173], PathTrust [174], NICE[175], Abdul-Rahman and Hailes[176], TrustMe[177], Advogato[178], Appleseed[179], PageRank[180], Josang [181], REGRET[183], Farag and Maheshwaran [184][185], Ramchurn[186], PTM[187], PET[188], Silaghi[189], eBay[190]	TUX-TMS is a flexible Reputation based Trust Management system which combines the merit of policy proof and reputation evaluation. TUX-TMS takes feedback from the entities and computes trustworthiness and reputation score of the entities and domains to enable decision making for performing the transactions

Chapter 5

Deployment, Testing and Validation of the Proposed Solutions

Previous chapters discussed in detail the design and implementation of the proposed interoperability and trust management solutions. Out of the three proposed interoperability solutions, one interoperability solution, TUX-INTERO and trust management solution, TUX-TMS have been integrated into Thapar University Extensible-Collaborative Interoperability and Trust Management Solution (TUX-CITMS) for testing purposes. TUX-CITMS and other two interoperability solutions: TUX-INTEROP and PIE have been deployed at Thapar University Campus, Patiala. The snapshots of the experimental results successfully exhibit secure and interoperable Grids. The results obtained have been presented and analyzed in this chapter.

5.1 TUX-CITMS: Thapar University Extensible-Collaborative Interoperability and Trust Management Solution: A Grid Testbed

To test and evaluate interoperability and trust management solutions designed, a testbed namely “TUX-CITMS” has been setup. It has been deployed at Thapar University Campus and mainly managed from the Centre of Excellence in Grid Computing. Keeping the intricacies of Grid hidden, this testbed provides an interactive and easy to use interface for users. The details of the experimentation are presented as a case study conducted on TUX-CITMS at Thapar University Campus.

This campus wide Grid setup is motivated by the need to allow easy access to researchers of the computational and data resources available across the departments and schools located widely in the Campus. This testbed spans across whole Thapar University Campus including entire academic area, some part of residential areas, utility area like library, cyber surfing labs, cafeteria etc. A Grid environment is set up using ASP.Net with C# Web Services. Database is created in Microsoft SQL 2005. Through this testbed, the impact of interoperability and trust management solutions for establishing secure Grids has been evaluated. Figure 5.1 shows the integration of TUX-INTERO and TUX-TMS to demonstrate interoperability and trust management in Grids where clouds depicting resources available, are managed by Resource Manager. Resource Manager extracts information about resources and stores in Logical Database which can be queried by users. Figure 5.2 shows the layout of TUX-CITMS, the Grid testbed, deployed to demonstrate the proposed solutions for interoperability and trust management.

Basically, there are three groups of stakeholder in this setup: the Service

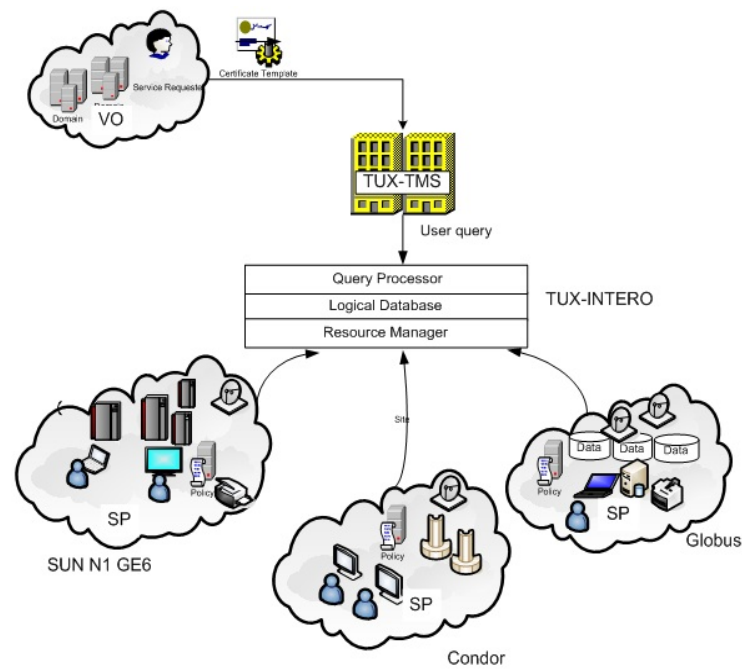


Figure 5.1: Integration of TUX-INTERO and TUX-TMS

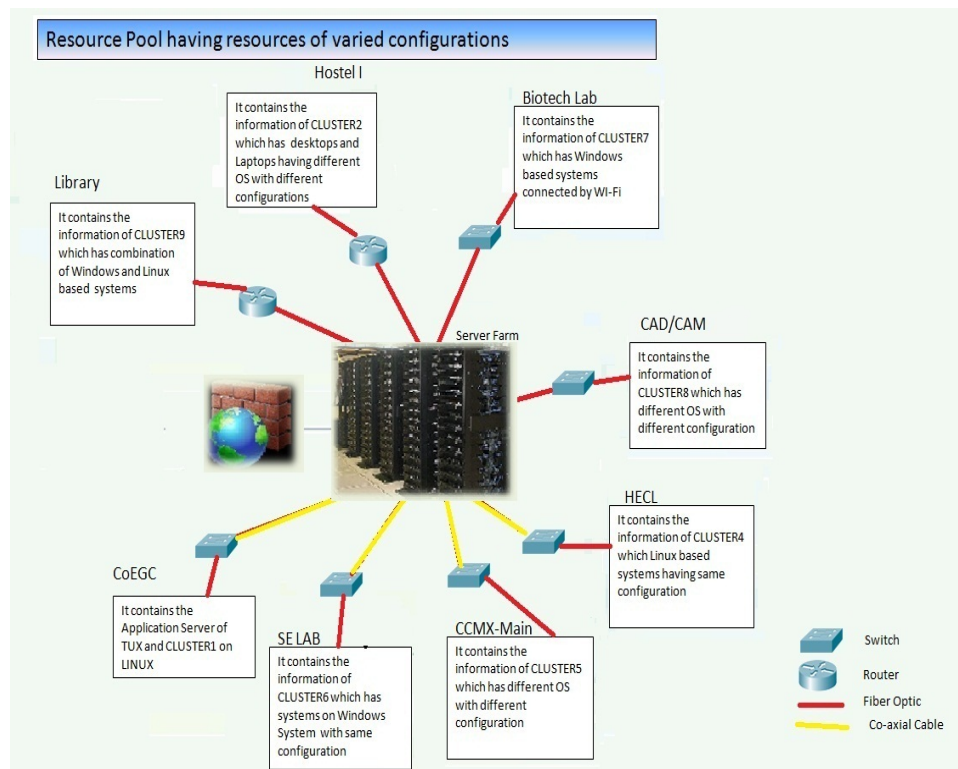


Figure 5.2: TUX-CITMS: Grid Testbed

Providers/Resources Owners, the Service Requesters/Resource Consumers and Administrator.

Service Providers/Resource Owners: Resource owners are the ones who have contributed resources to TUX-CITMS such as SE Lab, CCMX-Main, HECL Lab, CoEGC Lab, CAD/CAM, Library, Hostel I, Biotech lab etc.

Service Requesters/Resource Consumers: Resource Consumers are students and faculty of Thapar University who utilize the resources of TUX-CITMS. A Grid user account is needed to access TUX-CITMS. Information on account application is available at Centre of Excellence in Grid Computing.

Administrator: Administrator controls and monitors the activities of Service Provider and Service Requester to utilize resources available in TUX-CITMS by influencing only security related tasks rather than modifying trust scores and resource information.

The testbed allows users to make use of varied resources on the basis of their trustworthiness and reputation scores. Following criteria has been considered for the experimentation to provide interoperability and trust management solutions for establishing secure Grids.

- A user can request a single resource or multiple resources but can not get the same resource allotted until he has submitted required trust information for the resource in the form of feedback.
- A user should be able to request resource irrespective of underlying platform.
- The resources should be selected on the basis of trustworthiness, reputation and risk information of the Service Provider providing the resource for a particular context.

- The resources should be provided on the basis of trustworthiness, reputation and risk information of the Service Requester requesting the resource for a particular context.

The implemented solutions are tested with the testbed on the Live Network of Thapar University i.e. TUX-CITMS as also shown in Figure 5.2. The next section discusses evaluation parameters for TUX-CITMS.

5.1.1 Evaluation Parameters

The solutions have been evaluated based upon the evaluation criteria explained in Chapter 3 and Chapter 4. Description of these metrics is as follows:

- **Interoperability:** Interoperability means that the resources are accessible to Grid users irrespective of the hosting environments that they belong to. It aims at federating resource information from various information services available of resources hosted on varied platforms.
- **Trustworthiness:** Trustworthiness is the firm belief in the competency of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather it is subject to the entity's behavior and applies only within a specific context at a given time.
- **Reputation:** The reputation of an entity is an expectation of its behavior based on other entities observation or information about the entity's past behavior with respect to a specific context for a given period of time.
- **Risk Assessment:** Risk assessment checks the risk involved in transacting with a particular entity/domain.

- **Trust Context Factor:** At any time t , the purpose of using services of a Service Provider by a Service Requester can be termed as the trust context factor, which may vary from application purpose. The trust weights may be assigned and evaluated likewise. Sensitive applications involving confidential information storage and exchange is considered to have higher context factor than simply using print services.
- **Transaction Success Rate:** At any time t , the total number of successful transactions incurred is considered as Transaction Success Rate.

The next section discusses Experimental Details of TUX-CITMS.

UserID	Username	Password	FirstName	LastName	Affiliation	EmailID	City	State	Country	Contactno	Cont...	Usertype	CreatedDate	Rating	Status
1	sanjay	sanjay	Sanjay	Kumar	TU	sanjay@gmail.com	Patiala	Punjab	India	998888888	51 JP...	R	4/18/2010 6:19:...	3.00	Approved
2	shashi	siddhi	Shashi	Bhanwar	TU	shashi@thapar....	Patiala	Punjab	India	9988529211	A-2 B...	A	4/18/2010 6:35:...	0.00	Approved
3	meenakshi	meenakshi	Meenakshi	Verma	TU	mmalik@thapar....	Patiala	Punjab	India	9898989822	51 JP...	P	4/29/2010 12:5:...	5.00	Approved
4	pujita	pujita	Pujita	Sharma	TU	pujita@thapar.edu	Patiala	Punjab	India	9898989898	A-2 B...	P	4/29/2010 1:01:...	6.00	Approved
5	seemu	narayan	seemu	sharma	TU	seemu.sharma@...	Patiala	Punjab	India	9837102733	Thapar	R	7/26/2010 9:40:...	4.00	Approved
9	rani	rani	Rani	Sharma	ECED	rani@gmail.com	Patiala	Punjab	India	9982187329	Thap...	R	7/26/2010 7:23:...	2.00	Approved
10	samksha	samksha	Samiksha	Rajvansh	MECH	samksha@thapar...	Patiala	Punjab	India	827137432979	Thap...	P	7/26/2010 7:24:...	3.00	Approved
11	seema	seema	Seema	bhanwar	Thapar U...	seema.bhanwar...	Patiala	Punjab	India	9928344324...	I 303	P	8/5/2010 7:32:0...	4.00	Approved
12	gurpreet	gurpreet	Gurpreet	Khaira	TU	gurpreet@gmail...	Patiala	Punjab	India	9888960210	I350	P	8/5/2010 7:36:0...	5.00	Approved
13	amita	amita	Amita	Sharma	TU	amita@thapar.edu	Patiala	Punjab	India	9779933474	TU Fac	P	8/5/2010 7:38:0...	7.00	Approved
25	nidhi	nidhi	Nidhi	Garg	Thapar U...	nidhi.kavya@gm...	Patiala	Punjab	India	9417561400	CSED	P	8/8/2010 2:39:0...	8.00	Approved
26	ratinder	ratinder	Ratinder	Kaur	CSED	ratinder.kelley@...	Patiala	Punjab	India	9780021546	Thap...	P	8/8/2010 2:40:0...	5.00	Approved
27	rajni	rajni	Rajni	Aron	CSED	rajni3387@gmail...	Patiala	Punjab	India	9463433008	Thap...	P	8/8/2010 2:40:0...	6.00	Approved
28	komal	komal	Komal	Jalan	CSED	komal.jalan@tha...	Patiala	Punjab	India	9023657563	Thap...	P	8/8/2010 2:42:0...	3.00	Approved
29	lyotsana	lyotsana	Jyotsana	Sharma	CSED	lyotsana.lrsna...	Patiala	Punjab	India	9896076407	Thap...	P	8/8/2010 2:43:0...	4.00	Approved
30	inderpreet	inderpreet	Inderpreet	Singh	Thapar U...	inderpreet@tha...	Patiala	Punjab	India	9215535632	Thap...	P	8/8/2010 2:45:0...	2.00	Approved
31	zaraar	zaraar	Zarrar	Rana	CSED	zarrar@gmail.com	Patiala	Punjab	India	9780130706	Thap...	P	8/8/2010 2:46:0...	4.00	Approved
32	ajay	ajay	Ajay	Kumar	CSED	ajaycpp@gmail...	Patiala	Punjab	India	9891367546	Thap...	P	8/8/2010 2:47:0...	5.00	Approved
33	vandana	vandana	Vandana	Goel	CSED	vandana@thapa...	Patiala	Punjab	India	9781551703	CSED	P	8/8/2010 2:48:0...	6.00	Approved
34	tikka	tikka	tikka	singh	MECH	tikka@thapar.edu	Patiala	Punjab	India	98719372911	Thap...	P	8/8/2010 2:49:0...	7.00	Approved
35	yogesh	yogesh	Yogesh	Kumar	MECH	yogesh@thapar...	Patiala	Punjab	India	9823721371	Thap...	P	8/8/2010 2:50:0...	8.00	Approved
36	santosh	santosh	Santosh	Pathak	DBTES	santosh@thapar...	Patiala	Punjab	India	92486342462	Thap...	P	8/8/2010 2:51:0...	2.00	Approved
37	sourabh	sourabh	Sourabh	Deorah	CSED	sourabh.thapar...	Patiala	Punjab	India	9888915007	Thap...	P	8/8/2010 2:52:0...	3.00	Approved
38	shivank	shivank	shivank	arya	MECH	shivank.arya@g...	Patiala	Punjab	India	98623518234	Thap...	R	8/8/2010 2:53:0...	3.00	Approved
39	shriram	shriram	shriram	shriram	MECH	shriram@gmail.com	Patiala	Punjab	India	982312364463	Thap...	R	8/8/2010 8:33:0...	6.00	Approved
40	parneet	parneet	Parneet	Sachdeva	ECED	parneet@gmail...	Patiala	Punjab	India	98213632542	Thap...	R	8/8/2010 8:34:0...	7.00	Approved
41	surbhi	surbhi	Surbhi	Jain	LMT	surbhian@gmail...	Patiala	Punjab	India	99873242312	Thap...	R	8/8/2010 8:34:0...	8.00	Approved

Figure 5.3: User Records of TUX-CITMS

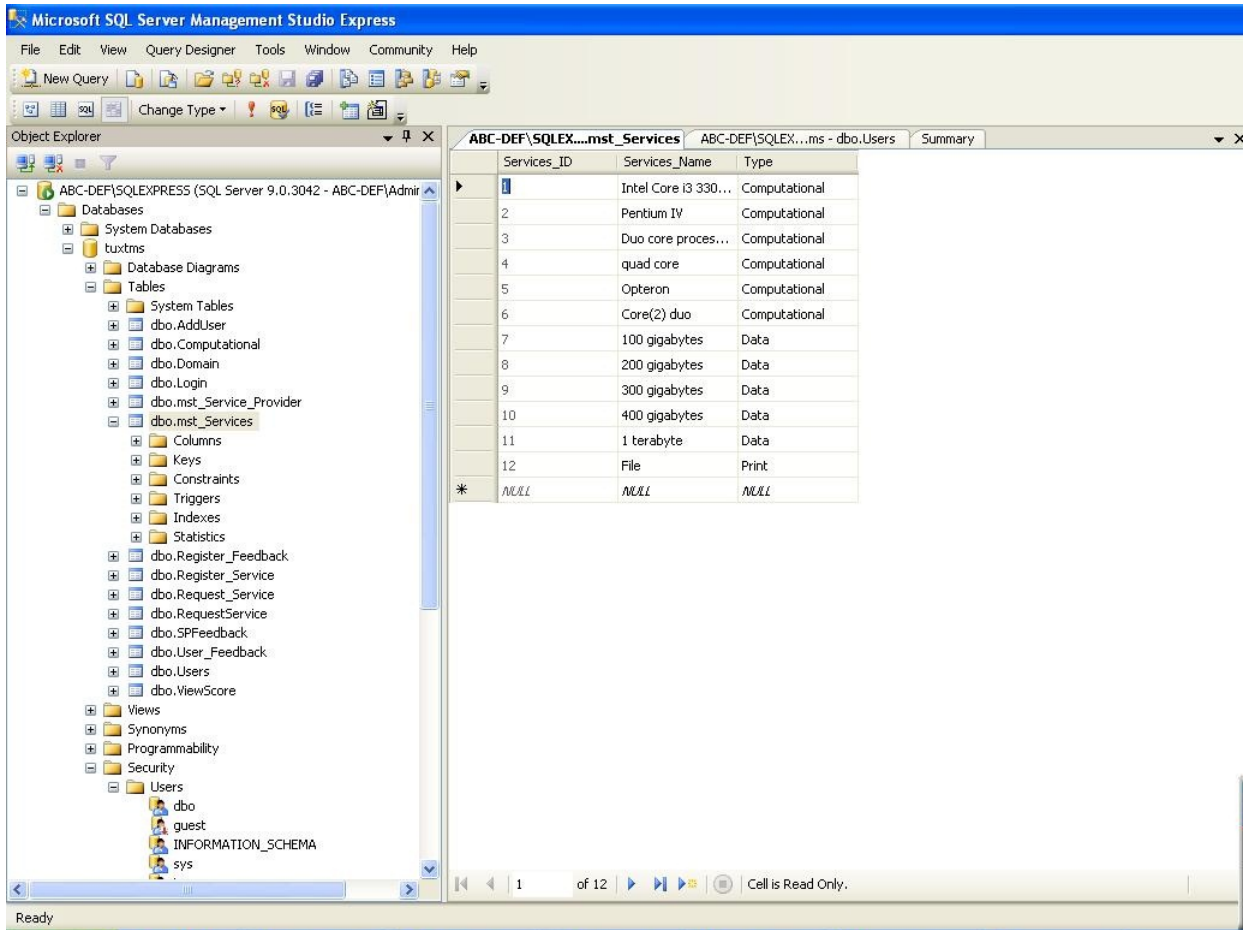


Figure 5.4: Services exposed to Service Requester in TUX-CITMS

5.1.2 Experimental Details

In this work, the implementation environment comprised of 30 Service Requesters from across Thapar University comprising of various departments, hostel and library, as discussed earlier. Figure 5.3 shows the snapshot information related to the Service Requesters and Service Providers in the form of User Records for TUX-CITMS in SQL database where the field UserID is unique to store trust related information. The other fields such as Username, Affiliation, Contact Information etc., contain basic user information. The field *Usertype* shows 'R' for Service Requester, 'P' for Service Provider and 'A' for Administrator. The field *Rating* shows score on the basis of trust score lying

in various zones as according to TUX-TMS's partitioning of users. The field "*Status*" shows whether the users have been approved by the Administrator after verification of credentials to become part of TUX-CITMS. However, the 20 Service Providers were located in Department of Computer Science and Engineering labs. Resources such as compute cycles, data storage and print facility have been exposed as services as shown in Figure 5.4. The Computational resources were chosen as node having configuration from this set {Intel core 2 duo, Opteron, Quad core and dual core}. The underlying platforms were chosen as node having installed one of the Operating System from this set {Windows Xp, Windows 7, Kubuntu, Suse Linux, Solaris 10, Slax, FedoraCore6, FedoraCore8}. The middleware were chosen as node having installed one of the middleware from this set {Globus Toolkit 4.2.1 (GT4), UNICORE 6}. The data services consists of data storage available from 100 gigabytes to 1 terabyte. Print service is also exposed. The users have been provided with security credentials which were generated through .Net tools. Trust database is created in SQL Server as TUX-DB. Access to resources/services is provided based on trust threshold as provided by TUX-TMS. For experimental evaluation, 30 Service Requesters across Thapar University were provided access to resources belonging to the 20 Service Providers located in three domains of the set {tiet.thapar.edu, lmt.thapar.edu, rnd.thapar.edu}. For each user (who may be a Service Requester or a Service Provider), trustworthiness and reputation scores are calculated by taking the feedback scores as input, after a significant number of successful transactions. A transactional database is created that, consists of sequence of transactions: $t = \{t_1, t_2, t_3, \dots, t_n\}$. Here, transaction t is composed of a Service Requester requesting a service from the Service Provider and then providing feedback and Service Provider providing access to resources and submitting feedback for Service Requester. Thereafter, an analysis of trust management metrics such as trustworthiness and repu-

Table 5.1: Grid Setup

Computational Service	Print Service	Data Storage	OS	Middleware	Service Requesters	Service Providers
Intel Core i3 330 Ghz, Pentium IV, Duo Core, Opteron, Quadcore, Core(2)duo	Print facility of Sharp ARM-205	100 gigabytes to 1 terabyte	Windows Xp, windows 7, Kubuntu, Suse Linux, Solaris 10, Slax, FedoraCore6, FedoraCore8	Globus Toolkit 4.2.1, Unicore 6	30 domains belonging to Thapar University	20 belonging to three domains tiet.thapar.edu, lmt.thapar.edu, rnd.thapar.edu from Thapar University

tation on the basis of trust context factor, trust and reputation decay, trust inheritance and risk assessment has been conducted. On the basis of the transactions that have taken place and feedback scores submitted, trustworthiness and reputation scores are stored for future usage purposes.

Different implementations have been tested on systems of different hardware, software platforms and middleware. The results clearly depicts that interoperability provides Grid as a single massive infrastructure to the user and reputation based trust management solutions establishes secure Grids and facilitate transactions between unknown users.

5.1.3 Deploying TUX-INTERO for providing interoperability across multiple platforms at Thapar University Campus

Firstly, a user is required to login to TUX-CITMS. An existing user is allowed to login through his credentials. However, a new user is requested to register themselves with the Grid using his credentials. After verifying the user identity, the administrator issues login details to the user through the authentication process. TUX-CITMS assumes three types of users: Administrator, Service Provider and Service Requester. TUX-CITMS makes use of Role Based Access Control (RBAC) which allows users to

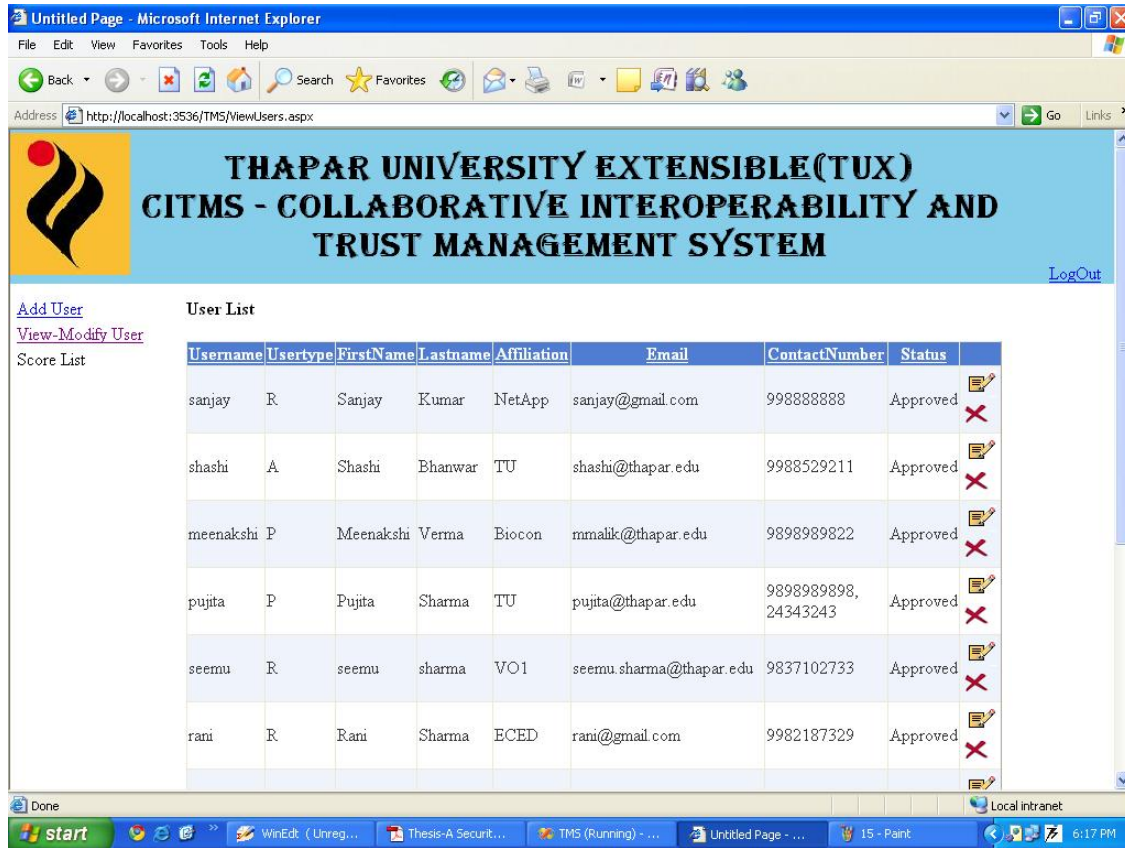


Figure 5.5: Administrator Screen in TUX-CITMS to View Users

login to their specific screen. After logging, depending upon the type of user, the screen is displayed. If a user is an administrator, he is allowed to Add and Modify users, as shown in Figure 5.5 and Figure 5.6. Once the users are approved by the Administrator then only they are allowed to become part of TUX-CITMS.

If a user registers as a Service Provider, he is allotted a Provider_ID and ID. The Service Provider is allowed to register services which are given unique ID as Resource_ID in Register_Service Table (discussed in Chapter 4).

If a user is a Service Requester, then the user is allowed to request resources which are available on the basis of trust threshold. A Service Requester can request resources from the resource list available, which is calculated on the basis of trust threshold. However, location of the resource is kept hidden from the Service Requester for secu-

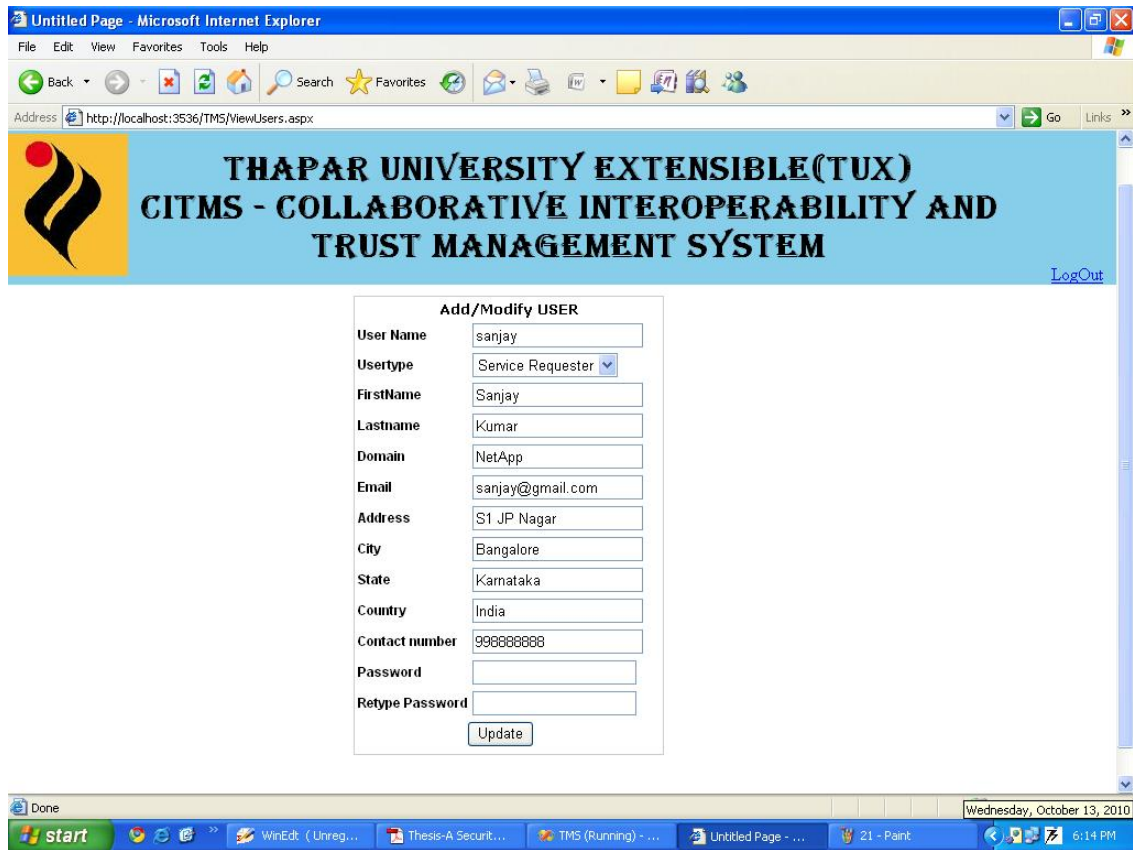


Figure 5.6: Administrator Screen in TUX-CITMS to Add/Modify Users

rity reasons. The resource discovery service used by TUX-CITMS acts as an online resource repository where the available Service Providers are shown to be available by querying the attribute Online as “TRUE” as shown in Figure 5.7. TUX-INTERO integrates nodes from Windows and Linux platforms using .Net Web Services. A desktop application is deployed onto these Windows and Linux nodes as shown in Figure 5.8, 5.9 and 5.10. This desktop application interacts with the TMServices’s ProviderService Web Service as shown in Figure 5.11 which is designed using .Net C# Web Service which is deployed on Windows Internet Information Services (IIS). The Web Service is invoked by Windows and Linux clients written in .Net and PHP, respectively. This information is send to the Application Server (TUX-CITMS). If a user login as a Service Provider then Provider_ID and Resource_ID are the IDs available to the individual

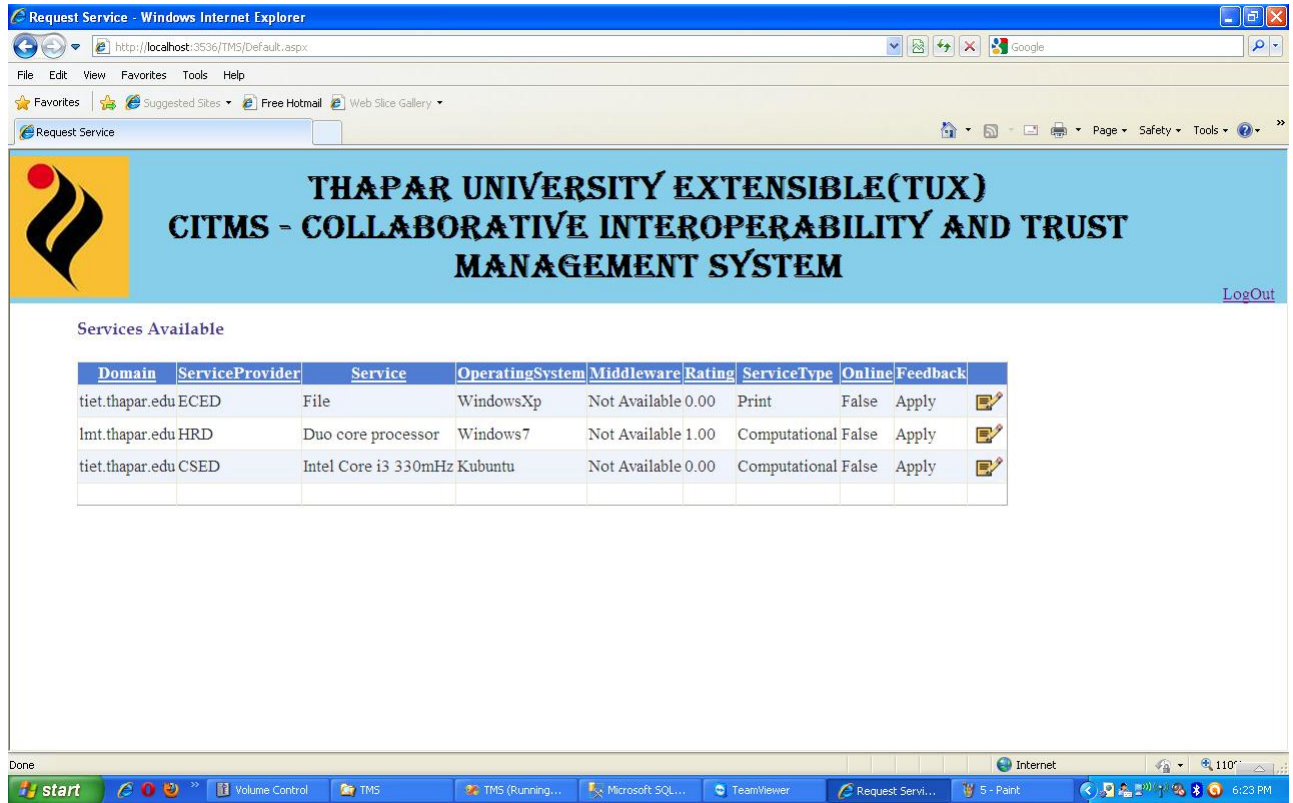


Figure 5.7: Services Available to the Service Requester

nodes deploying these desktop applications. These IDs are passed onto the Web Service clients to enable Service Provider to become part of the Grid as shown in Figure 5.11. Each Service Provider is allowed to register multiple services where Computational, Data and Print services has been exposed by TUX-CITMS as shown in Figure 5.12. This desktop application installed on Windows and Linux clients updates information of Service Provider in TUX-CITMS. The information of attribute “Online” as “True” shows the Service Provider is available. Further, a Service Provider can simply login into a system to advertise its resources to Grid. It makes use of Web Services to pass on information regarding its status using ServiceOnline() and ServiceOffline() web methods of ProviderService Web Service to the Application Server. Figure 5.13 depicts the ProviderService Web Service with ServiceOnline() and ServiceOffline() web methods.

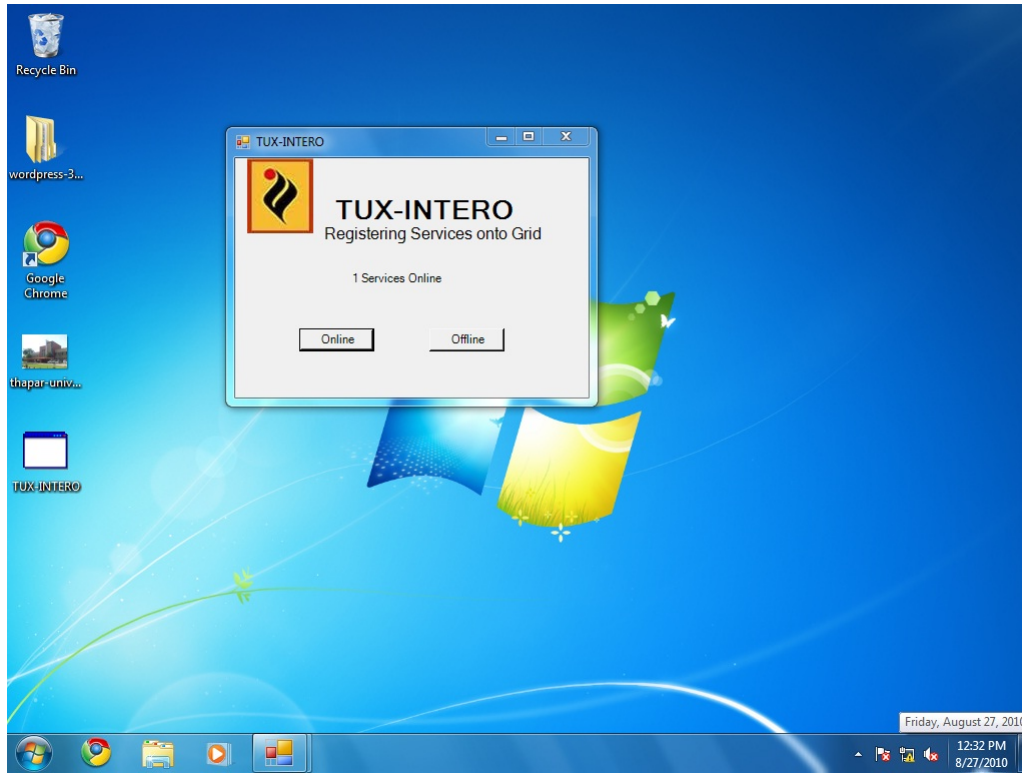


Figure 5.8: Desktop Application 1 of TUX-INTERO

TUX-INTERO is capable of providing interoperability across various platforms as it makes use of Web Services which are considered independent of the platform and can be invoked using client written in CORBA, Java, .Net etc. In TUX-INTERO, Grid resources are considered from various platforms to demonstrate interoperability. GT4 and UNICORE 6 is deployed for interoperability testing purposes on nodes. The interface between middleware specific resources (of GT4, UNICORE 6, Windows, Linux) and TUX-INTERO is implemented using a five step process of Extract, Parse, Check, Transform and Load, as explained in Chapter 3. The resource information is extracted from information services of various middleware such as Common Information Service (CIS) of Uniform Interface to Computing Resources 6 (UNICORE 6), Monitoring and Discovering Services 4 (MDS4) of GT4 and Web Services of Windows and Linux and transformed into GLUE 2.0 Schema and loaded into TUX-DB. The data transformation

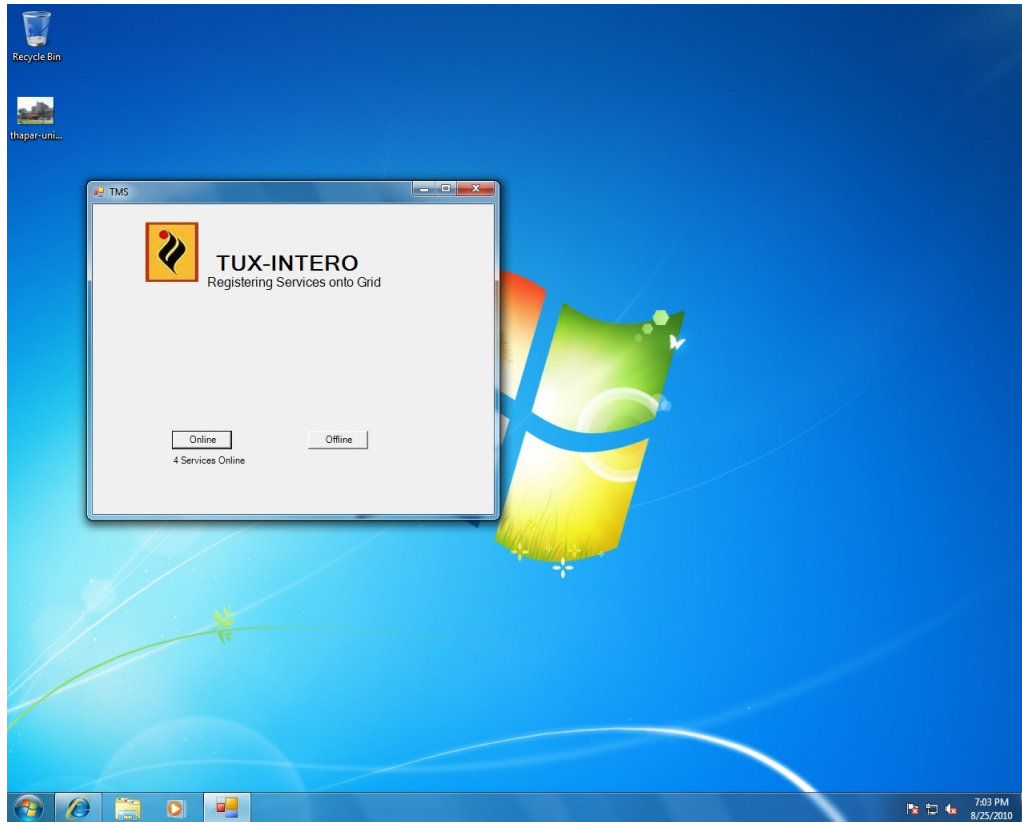


Figure 5.9: Desktop Application 2 of TUX-INTERO

is implemented using Extensible Stylesheet Language Transformations (XSLT), which is a popular language for processing XML data or transforming XML documents into other formats. This choice presented itself as most of the information services available provided their data in XML format.

CIS, the information system of the UNICORE 6 middleware, already uses the GLUE 2.0 Schema internally. Thus, the XSL Transformations for the UNICORE gateway merely checks the XML retrieved from the CIS and maps it to the respective SQL commands. No special rearrangement or interpretation of the data is needed. The monitoring system of Globus Toolkit 4 middleware, MDS4 is queried using a WSRF interface which provides output in XML format. As MDS4 is based on GLUE 1.1, the XSLT's of the MDS4 gateway are implemented to parse, check and map the fields with

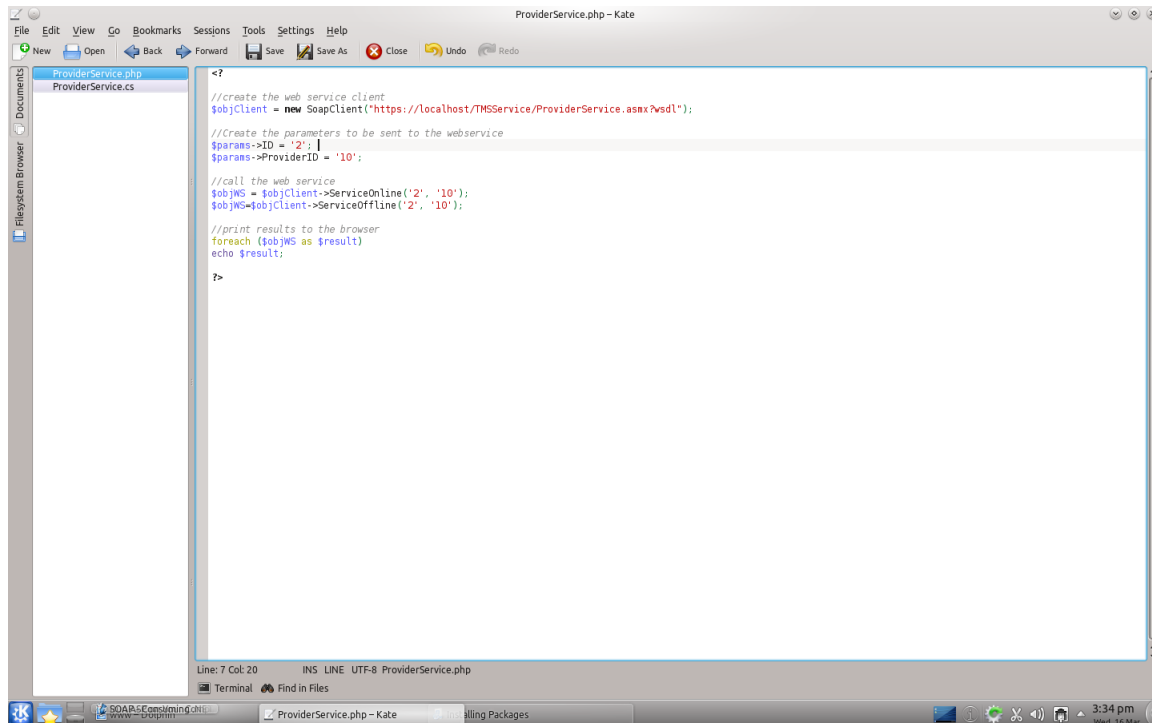


Figure 5.10: Invoking ASP. NET Web Services Using PHP

respective version of GLUE 2.0.

The setup using Windows and Linux was based completely on .Net Web Services. Therefore, the information regarding the resources was provided in form of XML which was transformed using XSLT into GLUE 2.0 for uniform representation.

TUX-DB can be queried for providing resource information which was done using SQL queries. Figure 5.7 shows the resources available to the end user independent of the underlying platforms.

5.1.4 Deploying TUX-TMS for establishing and evaluating Trust across Thapar University

Considering different sources for gathering trust information from (self experience, indirect experience, user/application trust requirements), each participant sorts out

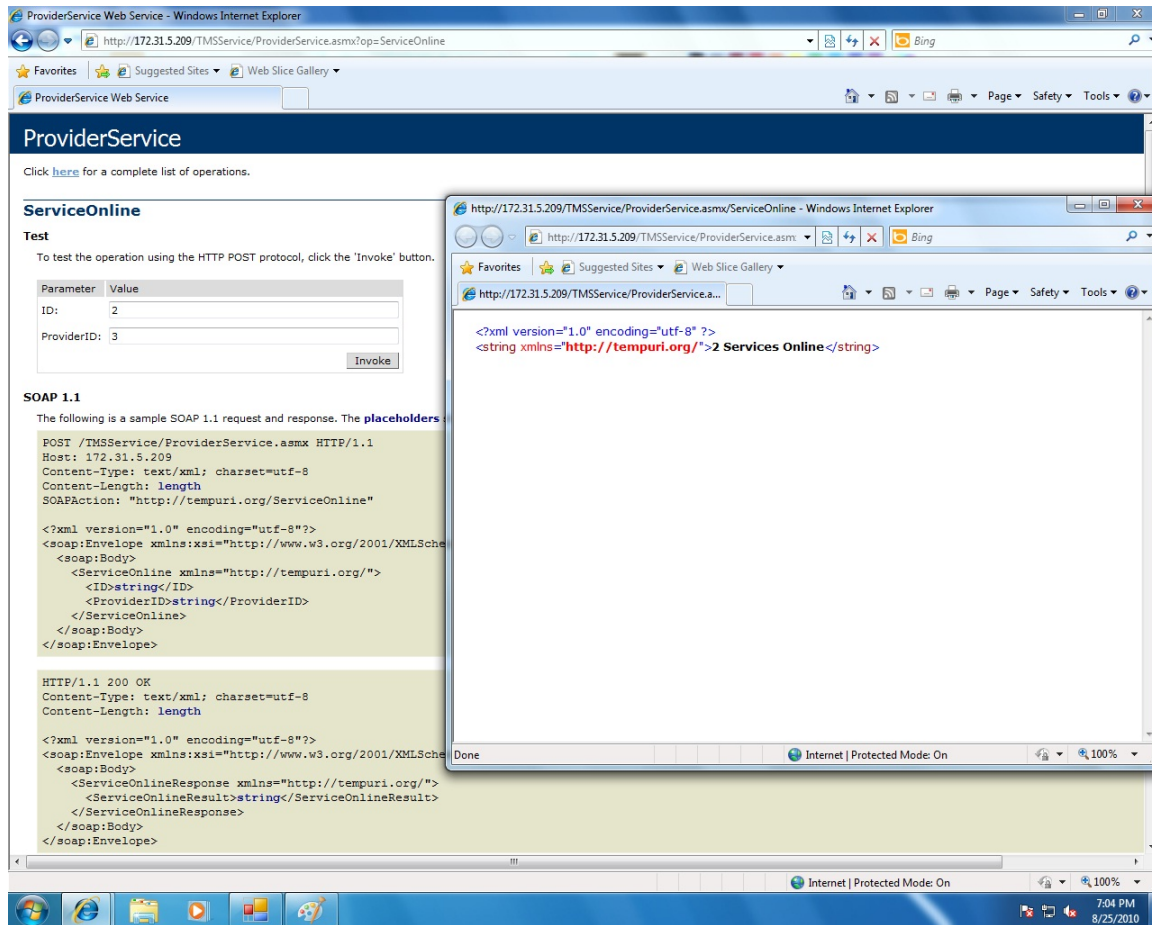


Figure 5.11: Web Services exposed to TUX-INTERO Desktop Application

the collaboration partners and starts interacting only with the “most trusted” of them. However, in TUX-TMS unknown partners are considered to interact.

As a new entity joins a domain, it initially has no Trustworthiness and Reputation values. A newcomer is allowed to interact with domain having certain threshold for services and it can then build its own trust values. In a Grid environment, entities are dynamic so, they can join and leave a virtual organization, at any time. Hence, a trust model for such an environment should have mechanisms for managing trust for such entities. When an entity joins a domain, it inherits the *Trust Inheritance* value from the domain, it is part of. If an entity is not gaining trust inheritance value then the introductory value is calculated on the basis of the domain introducing the

ID	Provider...	Domain	ServiceProvider	Service	Rating	ServiceType	Online	OperatingSystem	Middleware
2	3	biet.thapar.edu	CSED	Pentium IV	5.00	Computational	False	Kubuntu	Not Available
3	4	rnd.thapar.edu	RSLAB2	300 gigabytes	4.00	Data	True	Windows7	Unicore6
4	10	lmt.thapar.edu	LIBRARY	quad core	5.00	Computational	False	Windows7	Unicore6
5	11	rnd.thapar.edu	RSLAB2	Core(2) duo	5.00	Computational	True	FedoraCore6	Globus4.0.7
6	12	rnd.thapar.edu	RSLAB2	Duo core processor	5.00	Computational	True	FedoraCore6	Globus4.0.7
7	13	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	True	Windowsxp	Unicore6
8	25	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	True	Windowsxp	Unicore6
9	26	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Kubuntu	Not Available
10	27	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	FedoraCore10	Globus4.0.7
11	28	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	FedoraCore6	Globus4.0.7
12	29	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Windowsxp	Condor
13	30	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Windowsxp	Condor
14	31	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Windows7	Unicore6
15	32	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Windows7	Unicore
16	33	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Slax	Unicore6
17	34	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Slax	Unicore6
18	35	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Kubuntu	Not Available
19	36	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Kubuntu	Not Available
20	11	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	True	Windows7	Unicore6
21	12	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	True	Windowsxp	Condor
23	13	lmt.thapar.edu	HRD	200 gigabytes	4.00	Data	True	Windowsxp	Not Available
25	37	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Windowsxp	Not Available
26	37	lmt.thapar.edu	HRD	quad core	5.00	Computational	False	FedoraCore6	Not Available
27	37	rnd.thapar.edu	RSLAB2	200 gigabytes	5.00	Data	False	Windowsxp	Unicore6
28	37	lmt.thapar.edu	HRD	400 gigabytes	5.00	Data	False	Windowsxp	Not Available
29	35	biet.thapar.edu	CSED	300 gigabytes	5.00	Data	False	Windowsxp	Not Available
30	34	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Slax	Unicore6
31	34	biet.thapar.edu	CSED	Intel Core i3 330mhz	5.00	Computational	False	Slax	Unicore6

Figure 5.12: Service Records of the Services Available in TUX-CITMS

entity. The decay of the values of trustworthiness and reputation is initiated after a paramount number of successful transactions (which is considered minimum five here). The dynamic and fast changing Grid environment forces, only the recent values to be available to the end user for decision making purposes. Predictive information could lead to compromise of sensitive and powerful Grid resources. Risk assessment determines level of risk involved in transacting with domains having lower trust values. An entity for which both direct trust and recommendation trust exist, direct trust is given a higher weightage than the recommendation trust. Reputation of an entity is calculated from recommendation values submitted by the interacting entities. Any entity can give fraudulent recommendation, which is taken care of by aggregating all the recommendations given by a particular entity. False recommendation is discarded

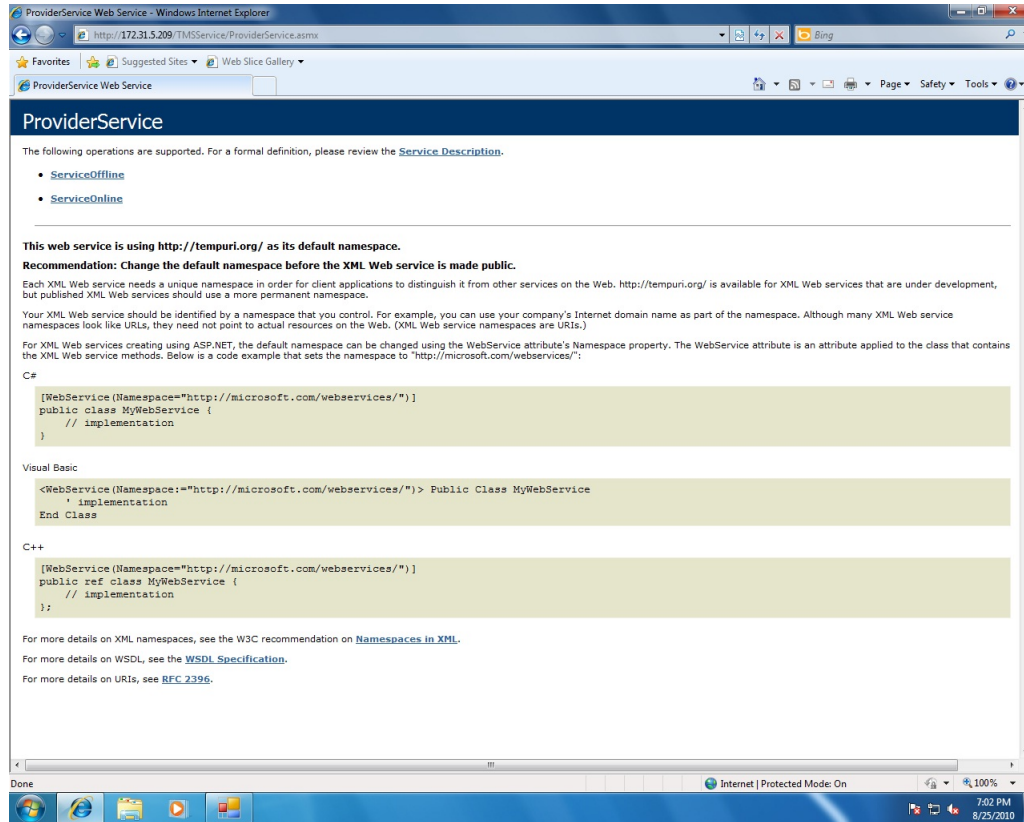


Figure 5.13: Invoking Web Services from TUX-INTERO Desktop Application

and penalty is imposed on domain providing it by making use of Intrusion Detection System (IDS) and Audit Trail, periodically.

For trust management, a Service Requester is required to access a resource. The resource can be accessed by selecting the button available for requesting the service as shown in Figure 5.14. Figure 5.14 shows the services provided to the entity having lower trustworthiness scores. However, the services provided to the entity having higher trustworthiness scores is more. In View Service Details screen in Figure 5.16, the Request Service button can be clicked to request the Service. The screen also shows the information about the Service Provider, which can be checked for trustworthiness and reputation information and risk involved in transacting with the Service Provider as shown in Figure 5.17. Figure 5.18 shows a Computational Service requested where

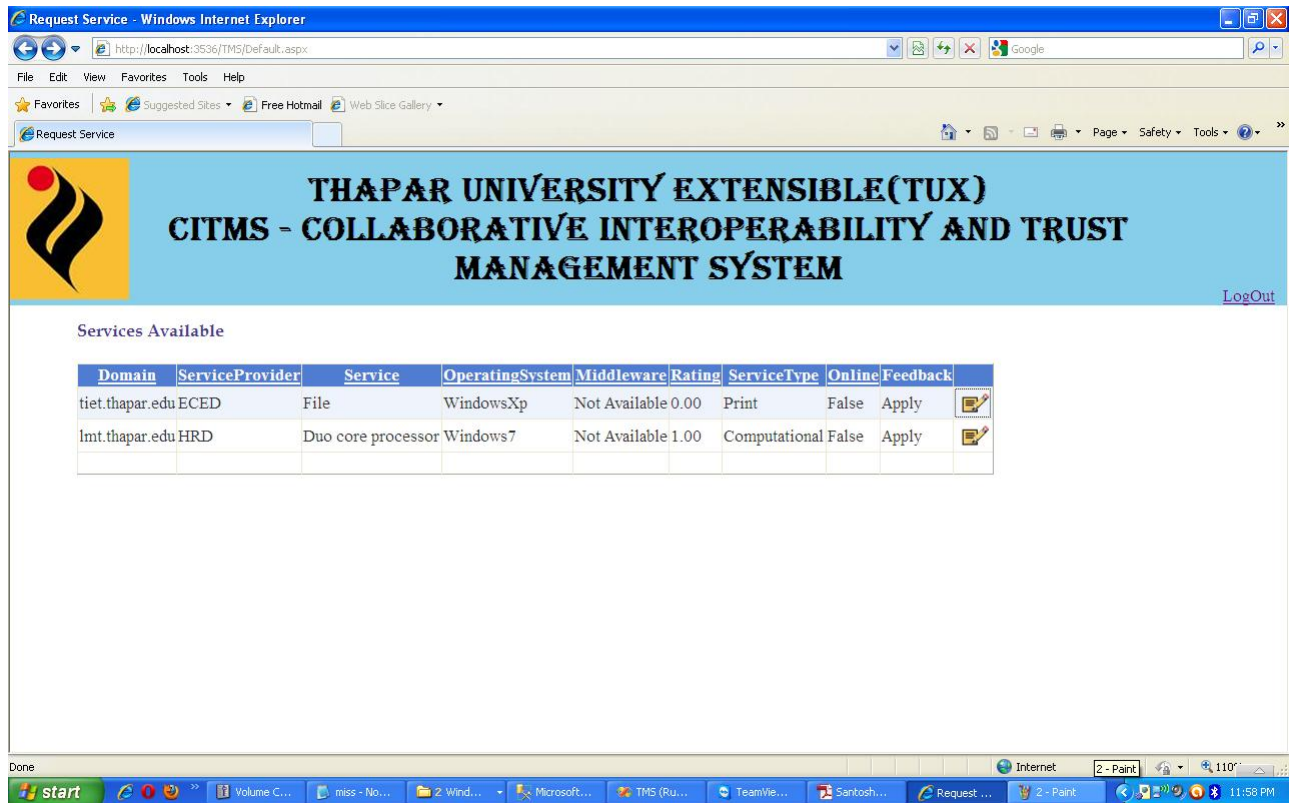


Figure 5.14: Result showing success of Test Case 1: Interoperability between nodes hosted in different versions of same Operating System

Service Requester can upload necessary details to be processed on requested resource. Once, the service is accessed and Service Requester receives the results of service requested such as Printed documents or results of applications. The resource can be only reutilized by submitting the feedback for the resource used as shown in Figure 5.19 and can be submitted using the form in Figure 5.20. The field *Feedback* in Figure 5.14 shows the status of Feedback information. If the feedback is submitted, the field shows the Feedback submitted and allows a Service Requester to request a Service again by closing service as shown in Figure 5.21. Once a feedback is submitted by the Service Requester, the Service Provider is also required to submit a Feedback for the Service Requester as shown in Figure 5.22. If the feedback is not submitted, the status information shown is Feedback Pending and the Service Requester is not allowed to

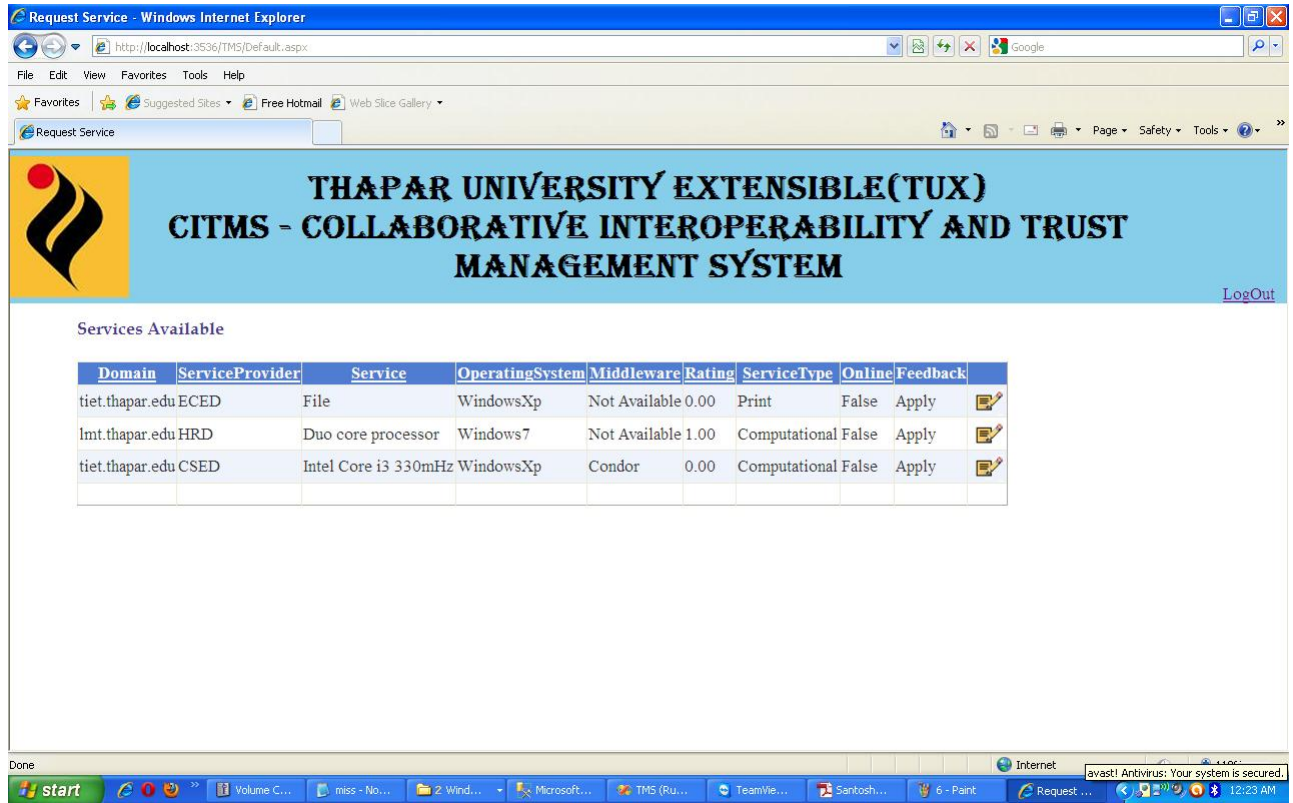


Figure 5.15: Result showing success of Test Case 2: Interoperability between systems hosted on different Operating Systems

access the service unless he has submitted the feedback. If a Service Requester has not submitted scores for five consecutive transaction then based on Classification of the service and the trustworthiness and reputation scores of the Service Requester, defaulters are penalized. The penalty imposed may be in form of reminders and then debarring Service Requester from using the service.

For trust management, the Service Provider can register the resources as shown in Figure 5.23. The Service Provider is requested to submit Feedback information for the services used for Service Requester as shown in Figure 5.24. If the feedback is not submitted for five consecutive transaction then based on Classification of the service and the trustworthiness and reputation scores of the Service Provider, defaulters are penalized. The penalty imposed may be in form of reminders and then debarring Service

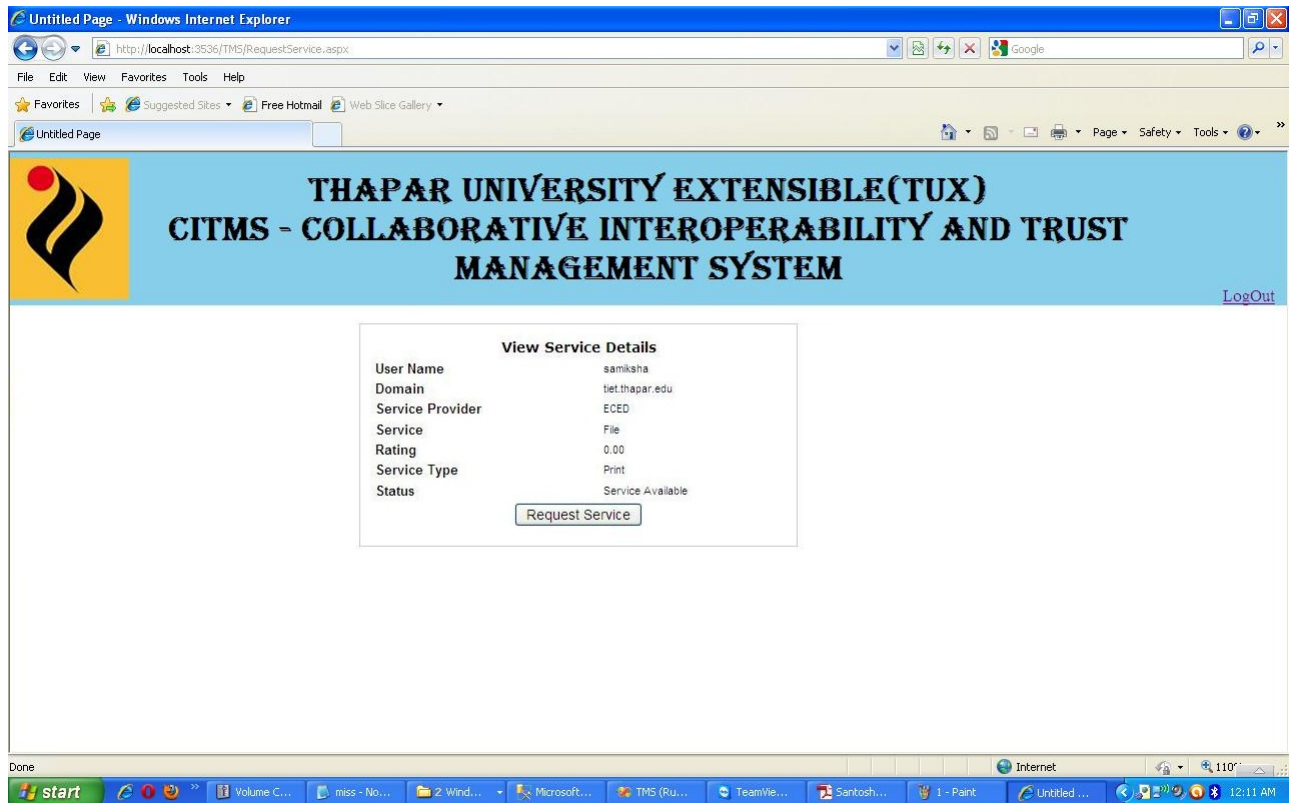


Figure 5.16: Requesting a Service by a Service Requester for accessing

Provider from providing the service. This two way process completes the transaction. These feedback scores are used to calculate trustworthiness and reputation of the users participating in the Grid. The scores of the feedback form contains questions which are meant for trustworthiness and recommendation of the end user who has utilized or provided the service. Each question carries scores which are aggregated and calculated for trustworthiness and reputation. This trust information is used to allow the Service Requester and Service Provider to become part of the Grid for future transactions. The next section presents Testing Details of the TUX-CITMS.

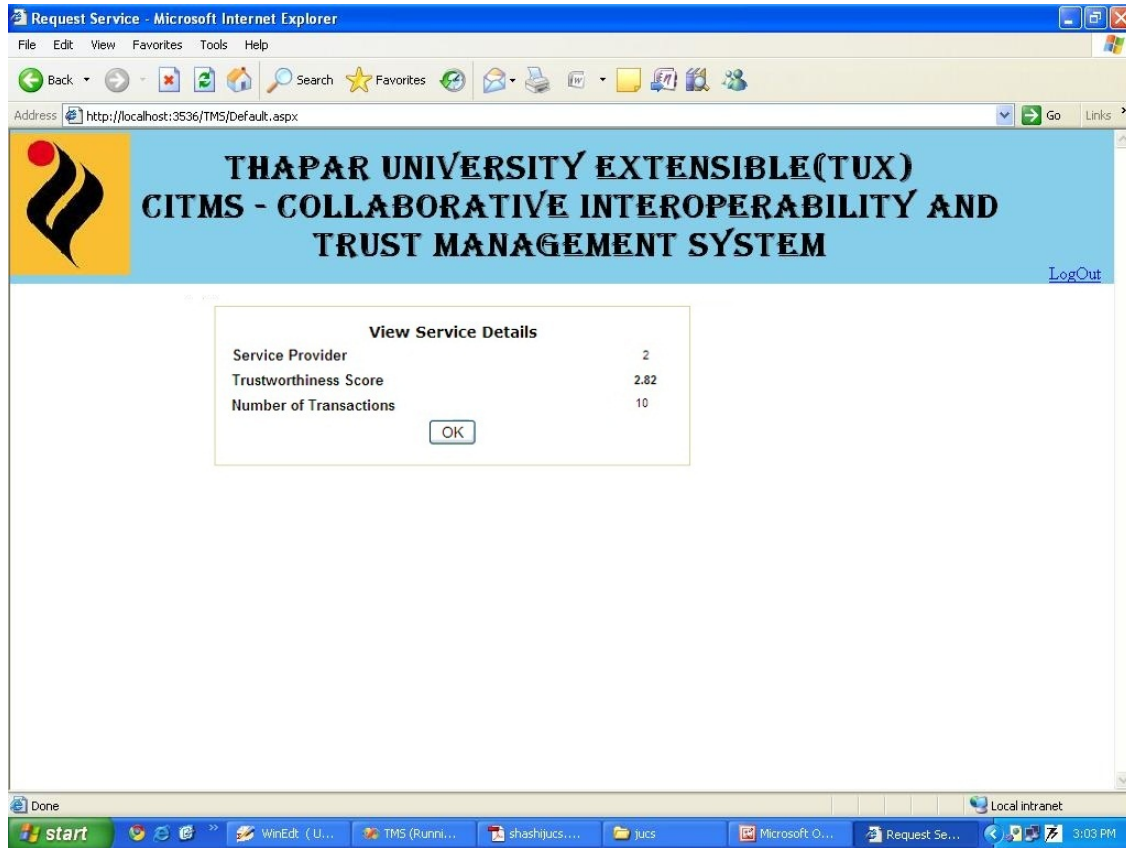


Figure 5.17: Result showing success of Test Case 12: View Trustworthiness Scores (Service Provider)

5.1.5 Testing Details

Testing is the process of examining an application to ensure it fulfills the requirements for which it was designed and meets quality expectations. The testbed details have been covered in previous section and platforms details are depicted in Figure 5.4. The testing has been conducted taking various test cases on TUX-CITMS as discussed below:

- (i) Test Case 1: In this test case, TUX-CITMS is tested for interoperability between nodes on different version of same operating system.

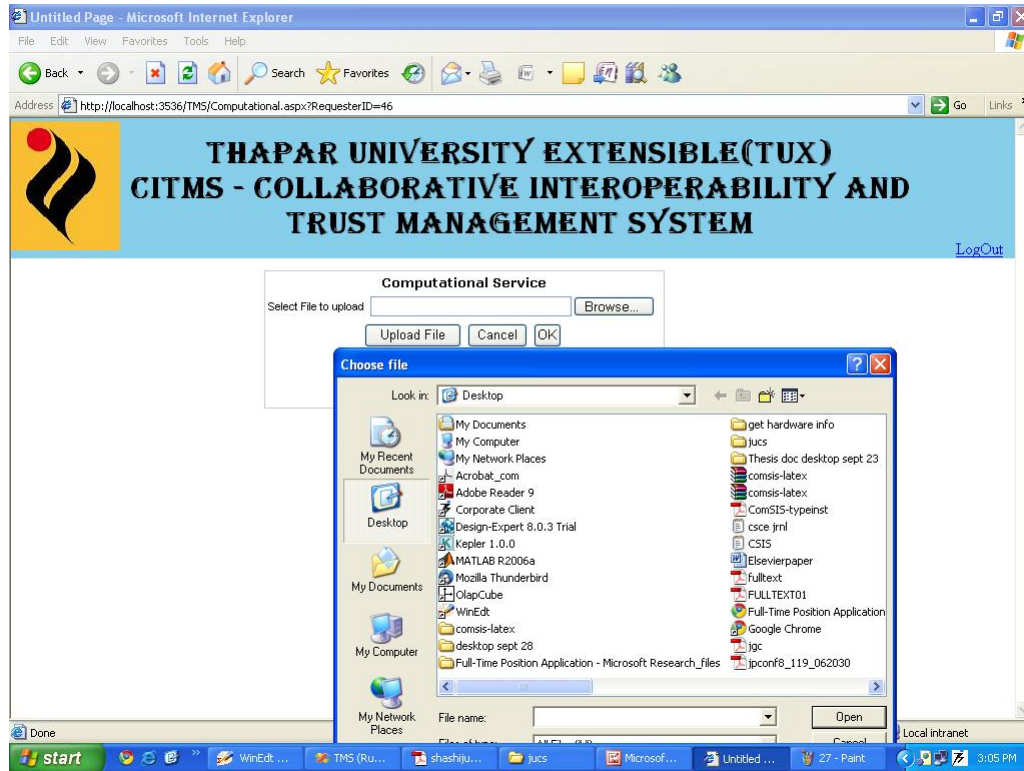


Figure 5.18: Accessing a Computational Service by a Service Requester

- Test case name: To test interoperability between nodes on different versions of same Operating System (OS).
- Feature to be tested: Operating System level interoperability between nodes on various h/w platforms and having same operating system installed.
- Test environment: required h/w and s/w to execute these test cases: Nodes running Windows Xp and Windows 7 operating system with .Net Client running on them. Web Server running with Internet Information Services (IIS) configured.
- Pre-condition necessary tasks to do before starting the test execution: ProviderService Web Service should be available to be invoked. TUX-DB should be running and connected with web server (TUX-CITMS).

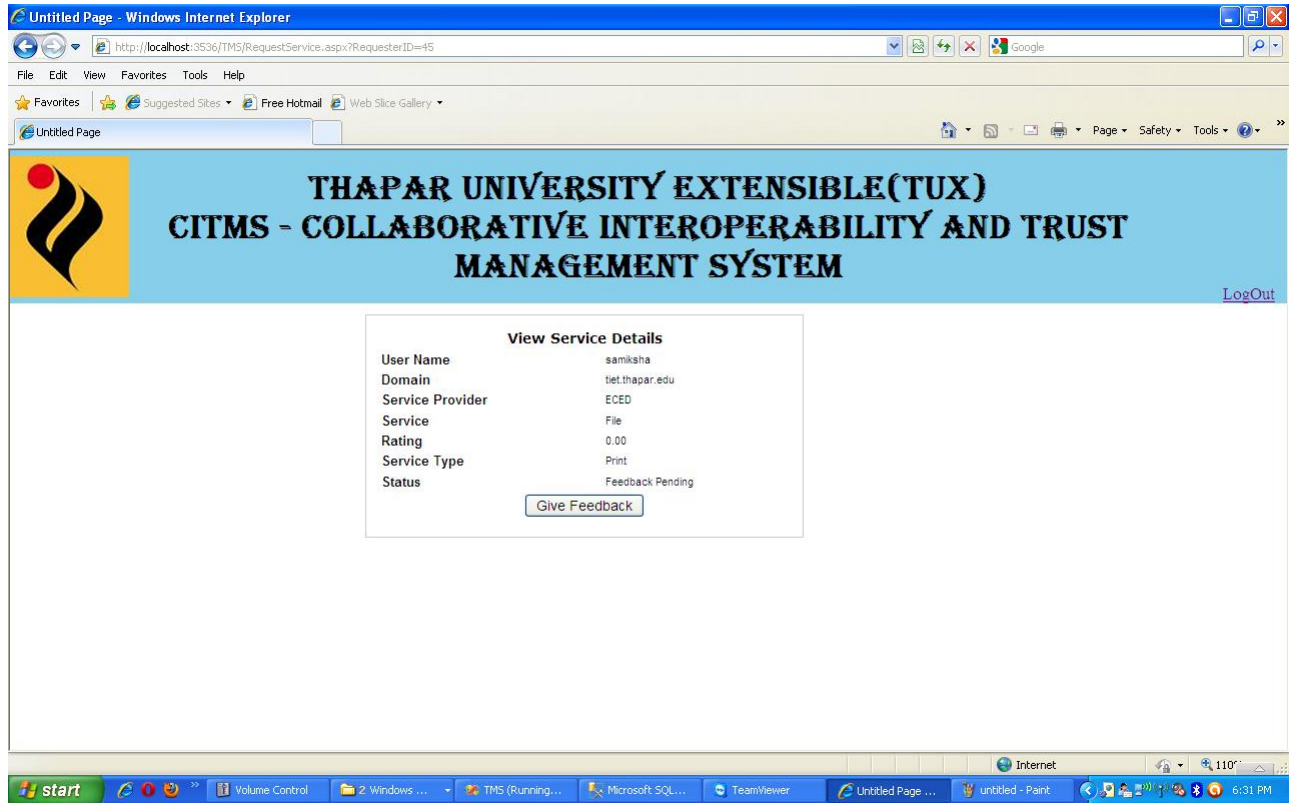


Figure 5.19: Submitting Feedback for the Services Accessed by Service Requester

- Test procedure: .Net Clients running on Windows Xp and Windows 7 Operating System invokes ProviderService Web Service located on local server and updates information to web sever (TUX-CITMS).
- I/P required: Web Service should be available to be invoked to update the information onto web server (TUX-CITMS). It connects resources on varied platforms to web server.
- Post conditions: A user can access resources hosted on different version of an operating system.
- O/P: The Figure 5.8 shows “1 Services Online” and Figure 5.9 shows “4 Services Online” to show that Windows System is communicate with Application Server and update its status as shown in Figure 5.25. Figure 5.14

**THAPAR UNIVERSITY EXTENSIBLE(TUX)
CITMS - COLLABORATIVE INTEROPERABILITY AND
TRUST MANAGEMENT SYSTEM**

[LogOut](#)

Feedback(Service Requestor)

- Overall, how satisfied were you with the product / service?
☐ Very good ☐ Satisfactory ☐ Bad ☐ Worst
- How would you rate the services provided by this Service Provider?
☐ Very good ☐ Satisfactory ☐ Bad ☐ Worst
- Thinking of similar products / services offered by other service provider in this domain, would you recommend this service provider to other?
☐ Highly recommended ☐ Recommended ☐ Not Recommended ☐ Dont Know
- Would you use this product / service in the future also?
☐ Definitely Yes ☐ Yes ☐ No ☐ Cant decide
- Would you recommend this product / service to colleagues or contacts within your domain?
☐ Highly recommended ☐ Recommended ☐ Not Recommended ☐ Dont Know

Figure 5.20: Feedback Form of Service Requester

shows the interoperability between nodes hosted on different version of same operating system. Interoperability between resources of same operating system but having different version as shown in Figure 5.14.

- Test case pass / fail criteria: Test case is “pass” if resources are available from different version of same operating system and case is “fail” if resources are not available different version of same operating system.
- (ii) Test Case 2: In this test case, TUX-CITMS is tested for interoperability between nodes on different operating system.
- Test case name: To test interoperability between nodes on different operating system.

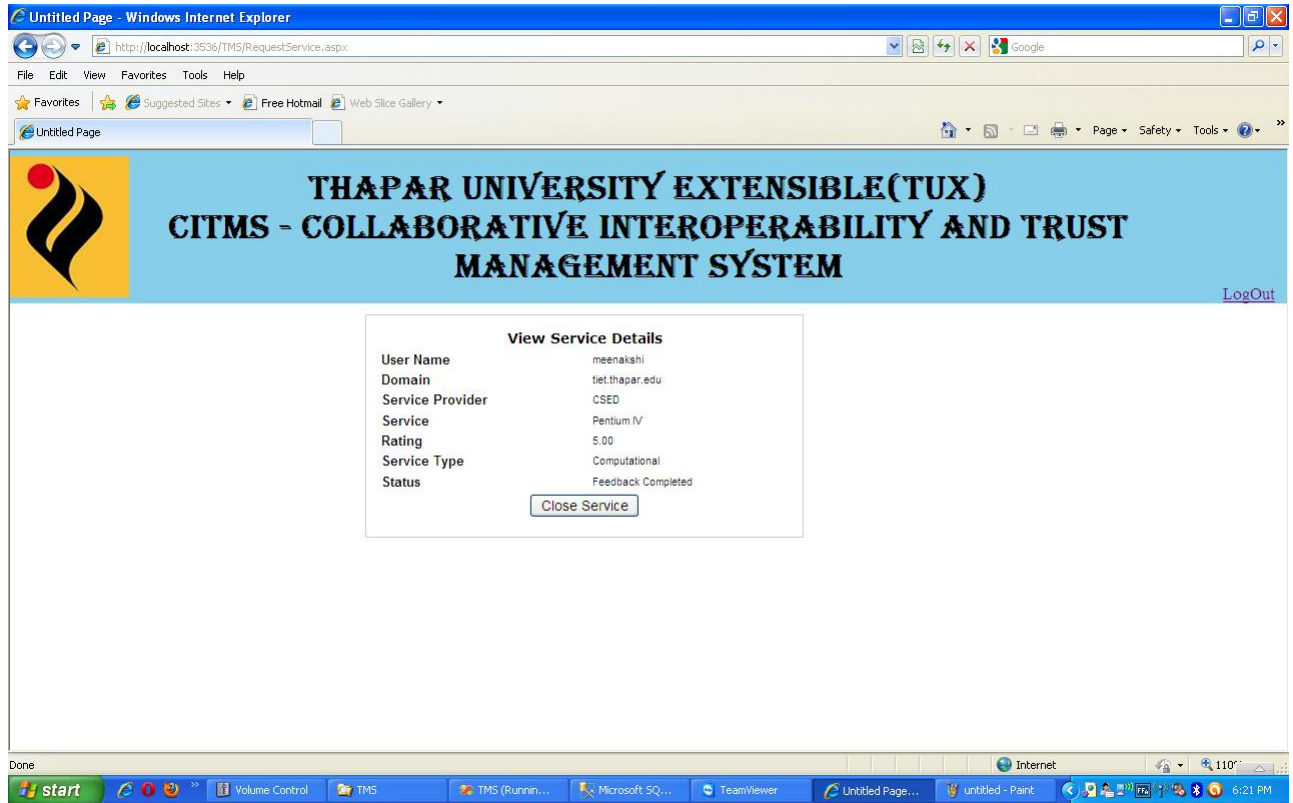


Figure 5.21: Closing a Service by Service Requester to use again

- Feature to be tested: Operating System level Interoperability between nodes on various h/w platforms and having different operating systems installed.
- Test environment: required h/w and s/w to execute these test cases: Nodes running Windows and Kubuntu operating system with .Net and PHP Client running on them, respectively. Web Server running with Internet Information Services (IIS) configured.
- Pre-condition necessary tasks to do before starting the test execution: ProviderService Web Service available to be invoked. TUX-DB should be running and connected with web server.
- Test procedure: .Net and PHP Clients running on Windows and Kubuntu Linux Operating System invokes ProviderService Web Service located on

THAPAR UNIVERSITY EXTENSIBLE (TUX)
CITMS - COLLABORATIVE INTEROPERABILITY AND TRUST MANAGEMENT SYSTEM

[LogOut](#)

Feedback (Service Provider)

1 Overall, how satisfied were you with the service requestor? ☐ Very good ☐ Satisfactory ☐ Bad ☐ Worst

2 How would you rate the trustworthiness of providing services to this service requestor? ☐ Very good ☐ Satisfactory ☐ Bad ☐ Worst

3 Thinking of similar products / services offered by other service provider of your domain, would you recommend this service requestor to use their services? ☐ Highly recommended ☐ Recommended ☐ Not Recommended ☐ Dont Know

4 Would you allow this service requestor to use your product / service in future? ☐ Definitely Yes ☐ Yes ☐ No ☐ Cant decide

5 Would you recommend the service requestor for using your other products / services? ☐ Highly recommended ☐ Recommended ☐ Not Recommended ☐ Dont Know

Figure 5.22: Feedback Form of Service Provider

local server.

- I/P required: .Net and PHP client should invoke Web Service to send information to web server.
- Post conditions: A user can access resources hosted on different operating system.
- O/P: The Figure 5.15 shows systems belonging to two different operating system. Windows System is able to communicate with Application Server and update its status. Likewise, Kubuntu based Linux client developed on PHP also invokes ProviderService Web Service and updates its status on TUX-CITMS.

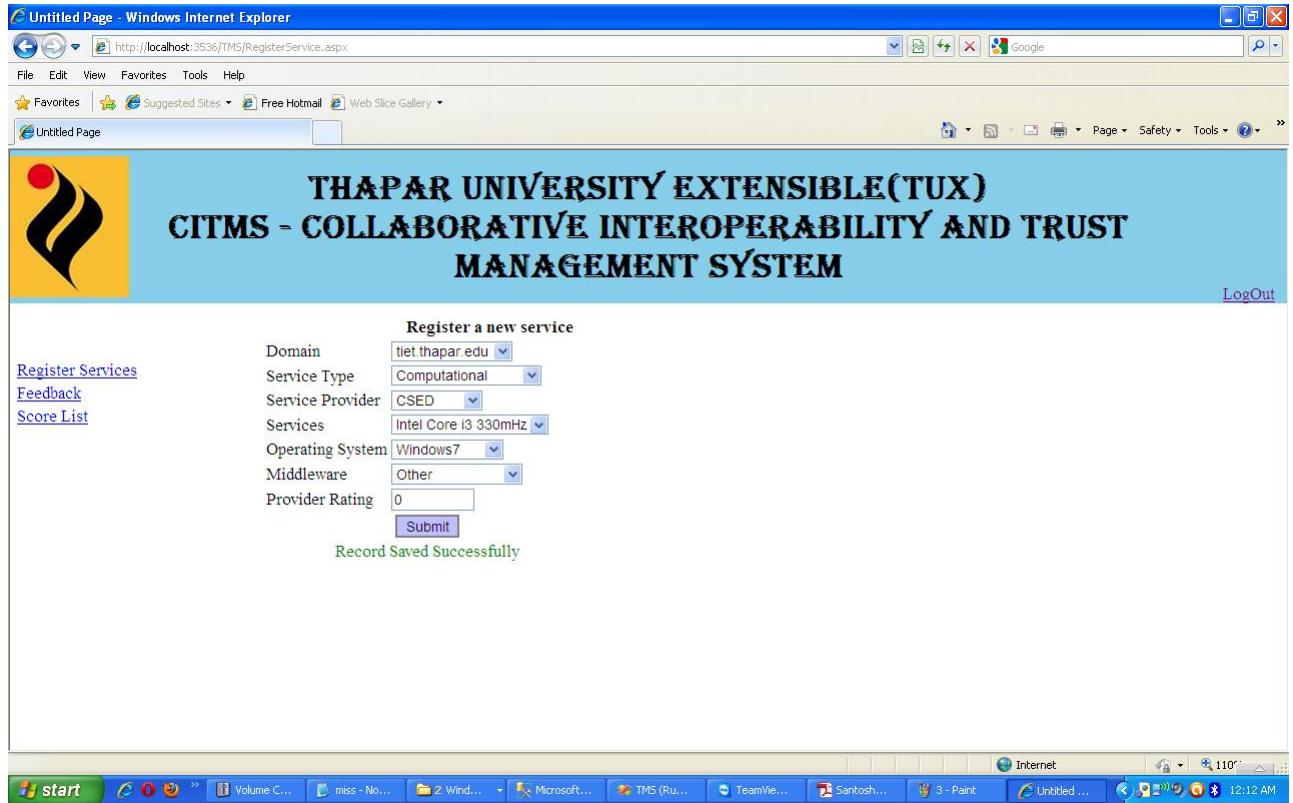


Figure 5.23: Registering a Service by a Service Provider in TUX-CITMS

- Test case pass / fail criteria: Test case is “pass” if resources are available from two different operating system and case is “fail” if resources are available from only one operating system.
- (iii) Test Case 3: In this test case, TUX-CITMS is tested for interoperability between nodes on various operating system and middleware.
- Test case name: To test Interoperability between nodes on different operating system and grid middleware.
 - Feature to be tested: Operating System and middleware level Interoperability between nodes on various grid middleware and operating system.

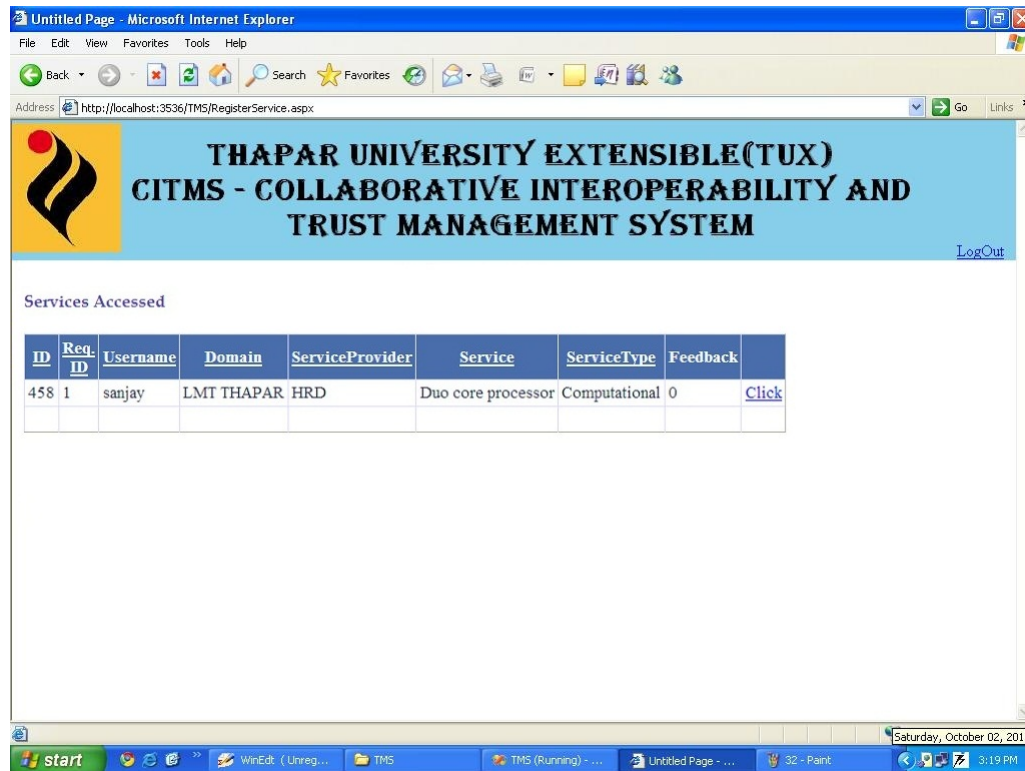


Figure 5.24: Service Provider Screen showing the services used by Service Requester

- Test environment: required h/w and s/w to execute these test cases: Nodes running Windows and Linux OS and GT4 and Unicore 6. Information Services of Globus and Unicore 6 i.e. MDS4 and CIS running. .Net Server running with Internet Information Services (IIS) configured. .Net and PHP Client running. Web Server and TUX-DB should be running and connected.
- Pre-condition necessary tasks to do before starting the test execution: Services enabled in TUX-CITMS of the Service Providers.
- Test procedure: Using Hyper SIGAR API extract information from various platforms. Transform into an XML style sheet and save the necessary details using SQL queries to TUX-CITMS.
- I/P required: Web Service enabled.

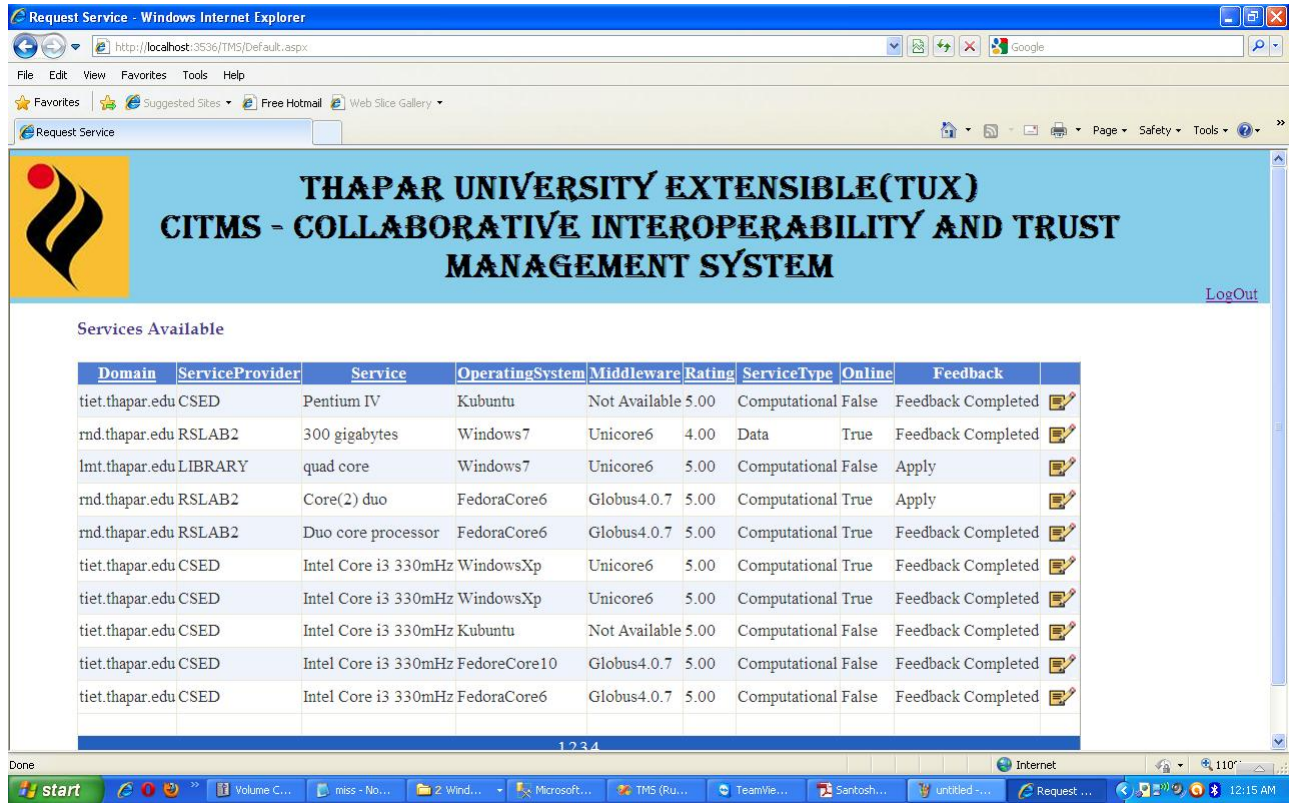


Figure 5.25: Result showing success of Test Case 3: Interoperability between systems hosted in different Operating Systems and Middleware

- Post conditions: Resources hosted on multiple operating system and middleware.
- O/P: The resource information is available to TUX-CITMS which is shown in Figure 5.25 which shows information of resources hosted on different operating system and middleware. The snapshots clearly depicts that TUX-CITMS provides system hosted on multiple platforms irrespective of the underlying architecture.
- Test case pass / fail criteria: Test case is “pass” if resources are available from various platforms, operating systems and middleware and case is “fail” if resources are available from only one platform, OS and middleware.

(iv) Test Case 4: In this test case, TUX-CITMS is tested for authentication.

- Test case name: To test and evaluate the usage of authentication in TUX-CITMS.
- Feature to be tested: Is there any benefit of authentication in a trust management system in a Grid Computing scenario.
- Test environment: required h/w and s/w to execute these test cases: Nodes running Windows and Linux OS and GT4 and Unicore 6. .Net Server running with Internet Information Services (IIS) configured. .Net and PHP Client running. Web Server running for Web Services and TUX-DB should be connected and running.
- Pre-condition necessary tasks to do before starting the test execution: Web server running and user credential details available.
- Test procedure: The proposed system handles authentication by requesting an end user to fill up New User registration form. Administrator of TUX-CITMS approves the end user to make use of the system by using authentication mechanisms such as Shared Key authentication and Public Key authentication for authenticating an end user. Then, an end user is allowed to use TUX-CITMS using his new credentials.
- I/P required: User Credentials.
- Post conditions: If the user credentials provided are authentic then access to TUX-CITMS is provided and user is allowed to access resources and utilize services.
- O/P: A user is authenticated as shown in Figure 5.15 and Figure 5.23.

- Test case pass / fail criteria: Test case is “pass” if authentic users are allowed to log in to TUX-CITMS and case is “fail” if credentials belonging to the user credentials are fake and authentication is provided.

(v) Test Case 5: In this test case, TUX-CITMS is tested for authorization.

- Test case name: To test and evaluate the usage of authorization in TUX-CITMS.
- Feature to be tested: Is there any benefit of authorization in a trust management system in a Grid Computing scenario.
- Test environment: required h/w and s/w to execute these test cases: Nodes running Windows and Linux OS and GT4 and Unicore 6. .Net Server running with Internet Information Services (IIS) configured. .Net and PHP Client running. Web Server running for Web Services. TUX-DB should be connected and running.
- Pre-condition necessary tasks to do before starting the test execution: Web server running and user should have log in details. Access Control Policies.
- Test procedure: TUX-CITMS uses Role Based Access Control (RBAC) and the three users in TUX-CITMS are provided access control, accordingly.

Login as an Admin: An administrator is the super user having rights to Add/Modify user. An administrator can check the credentials of the users registering and provide them with authentication information as per the authentication mechanism in process. An administrator cannot modify trust scores submitted by the Service Requester as well as Service Provider.

Registering as a Service Provider: A user can login into the TUX-CITMS system using his credentials. When a user logs in as a Service

Provider he is supposed to register some of his services for use by the Service Requester. A Service Provider can submit feedback information for the Service Requesters who have used their services.

Registering as a Service Requester: A user can login in as Service Requester by first registering into TUX-CITMS. A Service Requester is allowed to access a service provided by the Service Provider. A Service Requester can submit feedback information for the services used.

- I/P required: Role Based Access Control details.
- Post conditions: According to the access control policy user is allowed to use the set of services. Service Requester is allowed to request services and Service Provider is allowed to register services. Other tasks such as feedback submission and past transaction history is also provided to the user as per the access allowed.
- O/P: An administrator is allowed to Add/Modify user as per its access control as shown in Figure 5.5 and 5.6. A Service Requester is allowed to access resources as shown in Figure 5.15 and Service Provider is allowed to register resource as shown in Figure 5.23.
- Test case pass / fail criteria: Test case is “pass” if users are provided access as per their roles and case is “fail” if access is not allowed as per the roles allocated.

(vi) Test Case 6: In this test case, TUX-CITMS is tested for evolving trust in a newcomer.

- Test case name: To test and evaluate how trust is evolved in a newcomer in a trust management system.

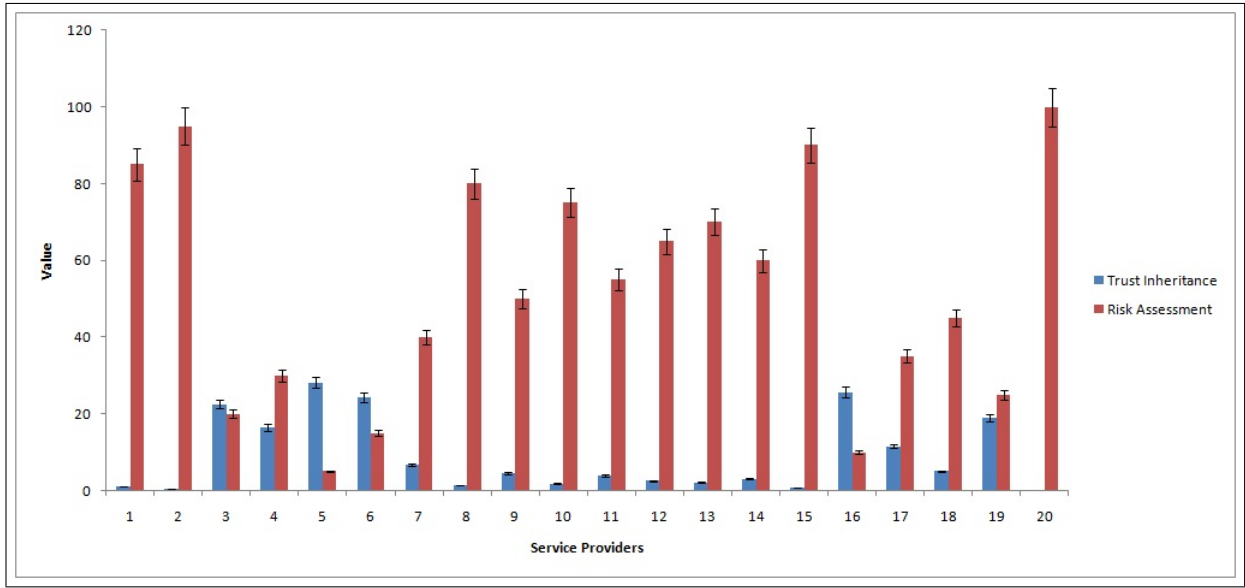


Figure 5.26: Trust Inheritance and Risk Assessment (Service Provider)

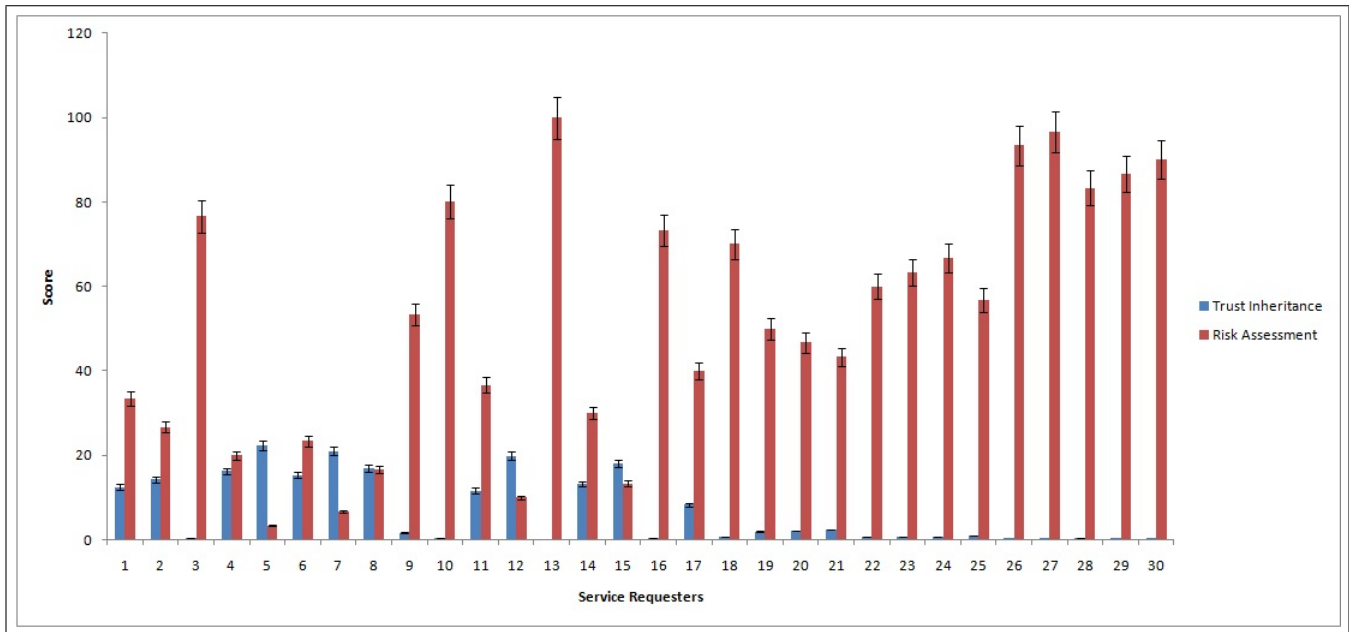


Figure 5.27: Trust Inheritance and Risk Assessment (Service Requester)

- Feature to be tested: Is there any benefit of evolving trust in a newcomer in a trust management system in a Grid Computing scenario.

- Test environment: required h/w and s/w to execute these test cases: Nodes running Windows and Linux OS and GT4 and Unicore 6. .Net Server running with Internet Information Services (IIS) configured. .Net and PHP Client running. Web Server running for Web Services. TUX-DB should be connected and running.
- Pre-condition necessary tasks to do before starting the test execution: TUX-DB and web server are connected and running.
- Test procedure: Using TUX-TMS to evolve trust as a newcomer. As a user login into TUX-CITMS, the trustworthiness, reputation, trust inheritance and risk assessment values are checked in TUX-DB. The proposed system handles assigning trust values to a newcomer by checking the inheritance level of the entity in a domain. A user may build trust values with the time by incurring number of transactions. A new user inherits trust values from the domains, it is part of. If an entity is part of a domain which has been transacting for a period of time then, trust values are inherited from the domain, else no value is assigned to the user.
 - Without trust inheritance: If an entity has entered as a newcomer and is not inheriting any trust value from any domain, then number of services allowed is very low.
 - With trust inheritance: If it inherits trust value, then the trust value inherited determines number of services available. Higher the trust value inherited more are the services available.
- I/P required: Trust related values of the users.

- Post conditions: If the user is new, then minimal set of services is provided to the user which are basically print services. If trust values are higher then higher set of services is provided.
- O/P: The results shown in Figure 5.26 and 5.27 shows the trust inherited by entities in a domain. Higher the trustworthiness and reputation score higher is the trust inherited by the entities becoming part of the Grid.
- Test case pass / fail criteria: Test case is “pass” if users are provided services as per their trust inheritance values or introductory value when they login as a newcomer and case is “fail” if services are not provided as per trust values of a newcomer.

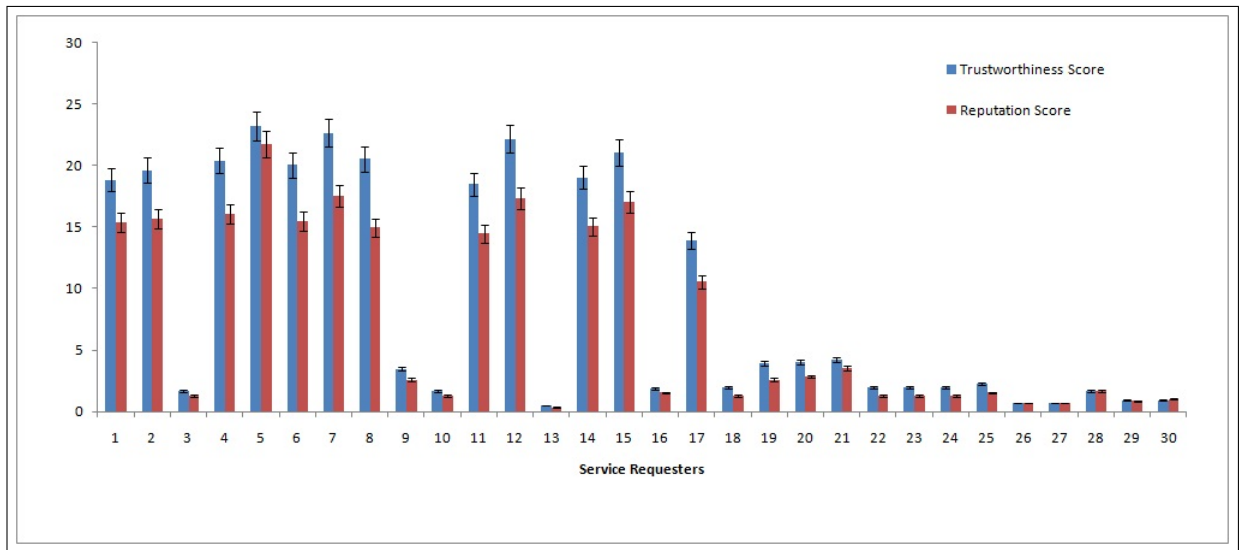


Figure 5.28: Trustworthiness and Reputation Score (Service Requester)

The Test environment remains same for Test Cases 7-16 as Test Case 5. However, Pre-condition for Test Cases from 7-11 elaborated below remains same as Test Case 6.

(vii) Test Case 7: In this test case, TUX-CITMS is tested for trustworthiness and reputation values of a Service Requester in a trust management system.

- Test case name: To test and evaluate the usage of trustworthiness and reputation values of a Service Requester in a trust management system in a Grid Computing scenario.
- Feature to be tested: Is there any benefit of trustworthiness and reputation values of a Service Requester in a trust management system in a Grid Computing scenario.
- Test procedure: The higher the trustworthiness and reputation values, more is the number of transactions and use of services in a Grid Computing scenario. As the number of transactions increase the trust value also increases marginally for accessing the resource without malicious purposes. The trustworthiness and reputation score is normalized by dividing it with the number of transactions with a particular domain. It discourages malicious domains to affect scores trustworthiness and reputation scores of malicious domains having an intent to increase values by incurring fraudulent transactions. The reputation of domain having higher trustworthiness and reputation trust values are considered to have higher reputation than others participant in a Grid. The reputation values are the recommendation values given by interacting domains.
- I/P required: Trustworthiness and reputation values of the users.
- Post conditions: If the user is Service Requester and has higher trustworthiness and reputation values then the number of services allowed to be accessed increases.

- O/P: The results shown in Figure 5.28 shows the trustworthiness and reputation scores of domains after on an average 50 transactions over a period of time. Domain 5, 7 and 12 shows higher trustworthiness and reputation scores for Service Requester. Higher the score of trustworthiness, more are the services allowed to the domain.
- Test case pass / fail criteria: Test case is “pass” if the user is Service Requester and has higher trustworthiness and reputation values and the number of services accessible increases and case is “fail” is the services accessible are less for higher trustworthiness and reputation values.

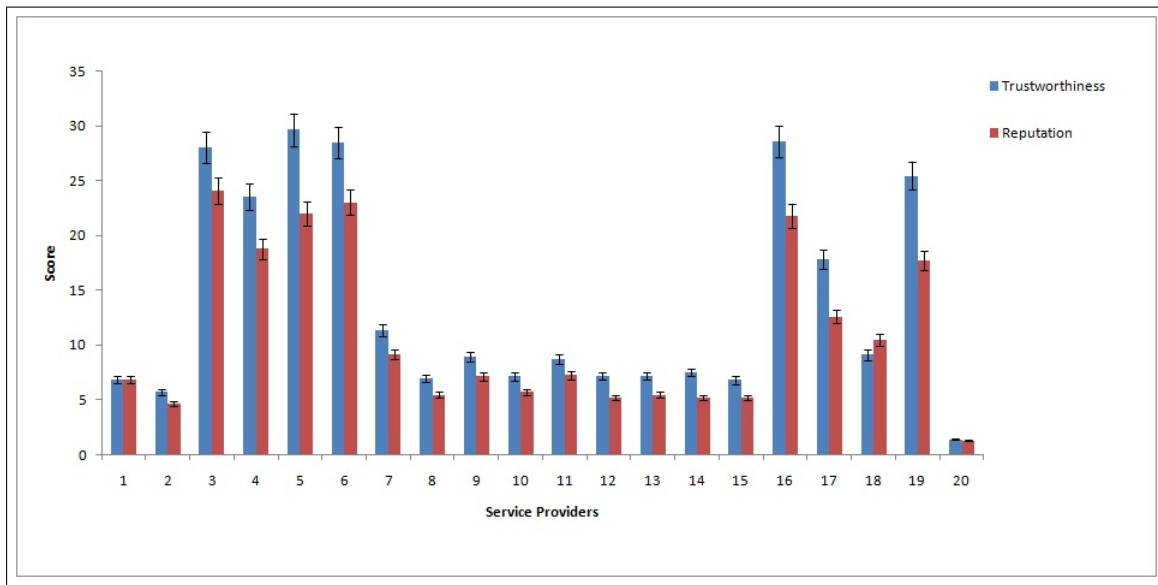


Figure 5.29: Trustworthiness and Reputation Score (Service Provider)

(viii) Test Case 8: In this test case, TUX-CITMS is tested for trustworthiness and reputation values of a Service Provider in a trust management system.

- Test case name: To test and evaluate the usage of trustworthiness and reputation values of a Service Provider in a trust management system in a Grid Computing scenario.

- Feature to be tested: Is there any benefit of trustworthiness and reputation values of a Service Provider in a trust management system in a Grid Computing scenario.
- Test procedure: The higher the trustworthiness and reputation values more is the number of transactions and use of services in a Grid Computing scenario. As the number of transactions increase the trust value also increases marginally for accessing the resource without malicious purposes. The trustworthiness and reputation score is normalize by dividing it with the number of transactions with a particular domain. It discourages malicious domains to affect trustworthiness and reputation scores of malicious domains having an intent to increase values by incurring fraudulent transactions. The reputation of domain having higher trustworthiness and reputation trust values are considered to have higher reputation than other participant in a Grid. The reputation values are the recommendation values given by interacting domains.
- I/P required: Trustworthiness and reputation values of the users.
- Post conditions: If the user is a Service Provider then the number of service usage increases leading to a high number of transactions.
- O/P: The results shown in Figure 5.29 shows the trustworthiness and reputation scores of a Service Provider after on an average 50 transactions over a period of time. Domain number 3, 5, 6 and 16 display higher trustworthiness scores and reputation for Service Provider.
- Test case pass / fail criteria: Test case is “pass” if the user is a Service Provider and the number of service usage increases leading to a high number of transactions and case is “fail” if service usage is lower.

TR	RP	Provider_ID	Transactions	Domains_Inter...	Tdecay	RDecay
7.37	7.35	3	11	10	0.68	0.73
5.7	4.65	4	9	9	0.63	0.52
28.03	24.1	10	43	16	0.65	1.51
23.53	18.8	11	36	24	0.65	0.78
29.63	22	12	47	24	0.63	0.92
28.47	23	13	45	16	0.63	1.44
11.37	9.15	25	19	12	0.6	0.76
6.93	5.45	26	12	12	0.58	0.45
8.93	7.15	27	14	12	0.64	0.6
7.13	5.7	28	12	12	0.59	0.47
8.67	7.25	29	14	12	0.62	0.6
7.2	5.2	30	12	12	0.6	0.43
7.2	5.45	31	12	12	0.6	0.45
7.5	5.2	32	12	12	0.63	0.43
6.8	5.2	33	12	12	0.57	0.43
31.33	24.5	34	50	30	0.63	0.82
17.8	12.6	35	32	12	0.56	1.05
9.1	10.5	36	17	9	0.54	1.17
25.43	17.7	37	47	12	0.54	1.47
1.4	1.3	62	2	2	0.7	0.65
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 5.30: Database results for Service Provider

(ix) Test Case 9: In this test case, TUX-CITMS is tested for trustworthiness and reputation values decay.

- Test case name: To test and evaluate the usage of trustworthiness and reputation values decay in a trust management system in a Grid Computing scenario.
- Feature to be tested: Is there any benefit of trustworthiness and reputation values decay in a trust management system in a Grid Computing scenario.
- Test procedure: When a user transacts over a period of time, the trust values are accumulated for the user which determines the actual values of users. However, trustworthiness and reputation values are decayed as a domain cannot be trusted at a same level after significant time has lapsed

TR	RP	Feedback_Rec...	Transactions	Domains_Inter...	Tdecay	Rdecay
19.47	15.9	1	31	26	0.63	0.61
19.63	15.7	5	29	26	0.68	0.6
1.7	1.3	9	3	3	0.57	0.43
20.4	16.05	38	33	31	0.62	0.52
23.17	21.7	39	34	34	0.68	0.64
20.03	15.5	40	35	35	0.57	0.44
22.63	17.5	41	39	39	0.58	0.45
20.53	14.95	42	39	39	0.53	0.38
3.469999999999...	2.6	43	6	6	0.58	0.43
1.7	1.3	44	3	3	0.57	0.43
18.47	14.45	45	34	34	0.54	0.42
22.17	17.35	46	37	37	0.6	0.47
0.5	0.35	47	1	1	0.5	0.35
19.03	15.05	48	31	28	0.61	0.54
21.03	17.05	49	33	32	0.64	0.53
1.9	1.55	50	3	3	0.63	0.52
13.87	10.55	51	24	23	0.58	0.46
1.930000000000...	1.3	52	3	3	0.64	0.43
3.93	2.6	55	6	6	0.66	0.43
4	2.85	56	6	6	0.67	0.48
4.23	3.55	57	7	6	0.6	0.59
1.97	1.3	58	3	3	0.66	0.43
1.97	1.3	59	3	3	0.66	0.43
1.97	1.3	60	3	3	0.66	0.43
2.23	1.55	61	3	3	0.74	0.52

Figure 5.31: Database results for Service Requester

between interactions. If a user is not participating into a Grid and not accessing the service, the trust values still are decayed to reflect dynamic nature of Grid. The decay of values make the rank different depending on the values obtained after decay after a period of time. The domains which are active participant remains in the current transaction details history of TUX-CITMS. The domains which are dormant and inactive for a period of time are slowly given lower ranks and have to establish their trust scores again to be a participant in a Grid Computing scenario.

- I/P required: Values of trustworthiness, reputation, number of transactions and number of domains interacted of the users.
- Post conditions: The domains which are active participants remains in a Grid Computing scenario and are ranked according to their values. However, the

domains which are dormant and inactive for a period of time are slowly given lower ranks and have to establish their trust scores again to be a participant in a Grid Computing scenario.

- O/P: The domains are ranked according to the values decayed after a long period of inactivity, the inactive domains are discarded from trust management system to maintain dynamicity of Grid Computing Environments. Figure 5.30 and 5.31 shows the trustworthiness decay (T_{decay}) and reputation decay (R_{decay}) values of the Service Requester and Service Provider.
 - Test case pass / fail criteria: Test case is “pass” if only active domains remain part of grid and case is “fail” if inactive domains are not discarded from the system.
- (x) Test Case 10: In this test case, TUX-CITMS is tested for risk involved in transacting with the domains.
- Test case name: To test and evaluate the usage of risk assessment in a trust management system in a Grid Computing scenario.
 - Feature to be tested: Is there any benefit of risk assessment in a trust management system in a Grid Computing scenario.
 - Test procedure: TUX-TMS is a ranking based model which determines the rank of domains on the basis of users entering the Grid, accessing it and submitting the experience in form of feedback for trustworthiness and reputation values. This ranking divides the domain list into quartile which are evaluated using risk assessment and risk impacts, as mentioned in previous chapter. The domains coming below the threshold of catastrophic and critical services are only allowed access to resources. There is a higher amount

of risk involved in transacting with domains having lower rank values.

- I/P required: Values of trustworthiness, reputation, number of transactions and number of domains interacted of the users.
- Post conditions: The domains which are having higher trustworthiness and reputation values are having lower risk in transacting. However, the domains having lower trustworthiness and reputation values have higher risk in transacting.
- O/P: Figure 5.26 and 5.27 also shows the risk assessment of domains. The domains having high trustworthiness and reputation values have low risk in transacting.
- Test case pass / fail criteria: Test case is “pass” if risk assessment provides the details of domain having higher risk in transacting and case is “fail” if risk assessment is not able to provide details of domains having higher risk in transacting.

(xi) Test Case 11: In this test case, TUX-CITMS is tested for identity trust.

- Test case name: To test and evaluate the usage of identity trust in a trust management system in a Grid Computing scenario.
- Feature to be tested: Is there any benefit of identity trust in a trust management system in a Grid Computing scenario.
- Test procedure: Identity trust is dependent on hard trust metrics of having established procedures for authentication and authorization in form of policies. Hard trust symbolizes the trust relationships that can be derived from the underlying cryptography based security mechanisms, such as digital certificates, signatures and cryptographic checksums. The process of

authentication of Identity Management System in TUX-TMS allows to establish identity trust onto domains providing genuine credentials. Intrusion Detection System (IDS) and Audit trails conducted periodically enables to root out malicious users.

- I/P required: Hard trust metrics which required established procedures for authentication and authorization in form of policies.
- Post conditions: Identity trust helps in authentication process to help provide access to authentic users and root out malicious users from the system as shown in Figure 5.3.
- O/P: Users are authenticated and can log into TUX-CITMS depending on the credentials available.
- Test case pass / fail criteria: Test case is “pass” if credentials provided are authentic and case is “fail” if credentials are fake and not verified by TUX-CITMS.

(xii) Test Case 12: In this test case, TUX-CITMS is tested for behaviour trust.

- Test case name: To test and evaluate the usage of behavior trust in a trust management system in a Grid Computing scenario.
- Feature to be tested: Is there any benefit of behaviour trust in a trust management system in a Grid Computing scenario.
- Pre-condition necessary tasks to do before starting the test execution: TUX-DB and web server are connected and running. Past transaction history.
- Test procedure: Behavior trust is dependent on the past behavior of the entity with other entities. It can be deduced using following methods:

- Knowing the reputation of domains: The past transaction history of the domain provides information about the reputation and behavior of the domain as shown in Figure 5.17. Figure 5.28 and Figure 5.29 shows that higher is the reputation of an entity, the better the entity has transaction history.
 - Transacting on the basis of policies: The domain can interact on the basis of policies under certain circumstances.
- I/P required: Past behaviour of the domain or entity becoming part of TUX-CITMS.
 - Post conditions: Behaviour trust helps in providing information about the reputation and trustworthiness of a domain/entity over a period of time so as to initiate a transaction with it. The higher the values of trustworthiness and reputation, the better the chances of transaction.
 - O/P: The past transaction details allow a user to decide with which user he can transact. The user can interact with a particular entity on the basis of trustworthiness and reputation information of the domain/entity. The Service Provider Past transactions can be checked using View Service Details in Figure 5.17.
 - Test case pass / fail criteria: Test case is “pass” if behaviour trust helps in establishing trust in a domain and case is “fail” if behaviour trust does not help in establishing trust in a domain/entity.
- (xiii) Test Case 13: In this test case, TUX-CITMS is tested for trust relationships.
- Test case name: To test and evaluate the usage of trust relationships in a trust management system in a Grid Computing scenario.

- Feature to be tested: Is there any benefit of trust relationships in a trust management system in a Grid Computing scenario.
- Pre-condition necessary tasks to do before starting the test execution: TUX-DB and web server are connected and running. Past transaction history.
- Test procedure: Trust relationships are necessary to determine the behavior of an entity while transacting in open distributed environments such as Grid. The domains rely on direct relationship behavior of an entity rather than relying on recommended trust. If an entity has behaved well in indirect trust relationship, the scores are still considered and domains are rewarded with incentives. However, direct trust is given more weightage than recommended trust. The trust relationships considered in TUX-TMS are:

- Direct trust
- Indirect trust
- Recommended trust

Trust relationships such as direct trust, recommended trust and indirect trust help to confide in the authenticity of domains while transacting and using past values. In case of domains having same trustworthiness and reputation values the domains having higher values in trust relationships are given preference. Indirect trust value is considered for exclusive cases where the domain has behaved good in case of unknown transacting domains.

- I/P required: Trust relationship information of the domain where transaction has taken place.
- Post conditions: A direct trust relationship values is given higher weightage over a recommended trust value.

- O/P: The relationship helps a user be it service provider or service consumer to only interact with trustworthy domains by checking the values obtained. The Past transaction history presents information about trust relationship to the Administrator which helps in ranking domains in case of same trustworthiness and reputation scores of several domains.
 - Test case pass / fail criteria: Test case is “pass” if trust relationship values are able to determine trustworthy users and case is “fail” if trust relationship values are not able to determine trustworthy users.
- (xiv) Test Case 14: In this test case, TUX-CITMS is tested for false recommendation.
- Test case name: To test and evaluate the usage of considering false recommendation in a trust management system in a Grid Computing scenario.
 - Feature to be tested: Is there any benefit of considering false recommendation in a trust management system in a Grid Computing scenario.
 - Pre-condition necessary tasks to do before starting the test execution: TUX-DB and web server are connected and running. Past transaction history and feedback information.
 - Test procedure: Audit trails of the feedback information provided are conducted periodically to find if any malicious domain is trying to increase trustworthiness and reputation values of domains malicious intents. The users are authenticated using Identity Management System (IMS) component of TUX-TMS and the feedback is analyzed periodically using Audit Trails to assess the false feedback submitted by the Service Requesters. Statistical methods are used in IDS and Audit trails to check the feedback variation and malicious domains are reported.

- I/P required: Audit trails information for the feedback provided for Service Requester as well as Service Provider.
- Post conditions: Malicious domains are rooted out of the TUX-CITMS so as to secure grid resources from attack.
- O/P: The audit trails details root out the users using TUX-CITMS for malicious purposes by cross verifying the data provided. Figure 5.2 shows the frequency of threats detected during the audits conducted for security threats in TUX-CITMS.
- Test case pass / fail criteria: Test case is “pass” if audit trails conducted for false recommendation analysis traces malicious domains and case is “fail” if if audit trails conducted for false recommendation analysis does not trace malicious domains.

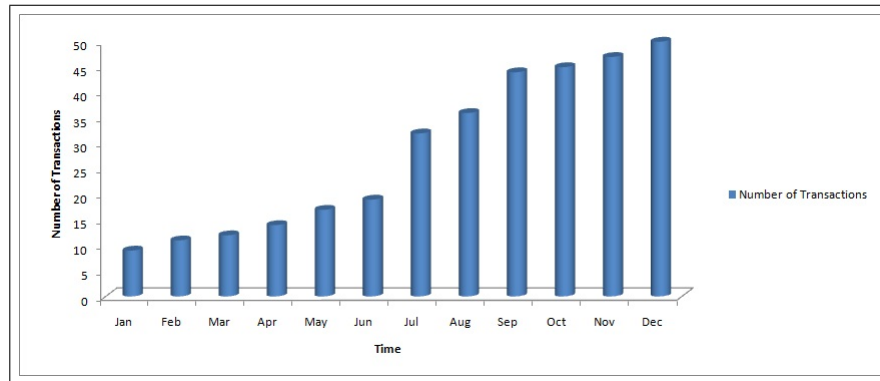


Figure 5.32: Result showing success of Test Case 15: Number of transactions were increased after deploying TUX-TMS into Grid Computing Environments

(xv) Test Case 15: In this test case, TUX-CITMS is tested for the usage of trust management system to secure grids.

- Test case name: To test and evaluate the usage of a trust management system in a Grid Computing scenario.

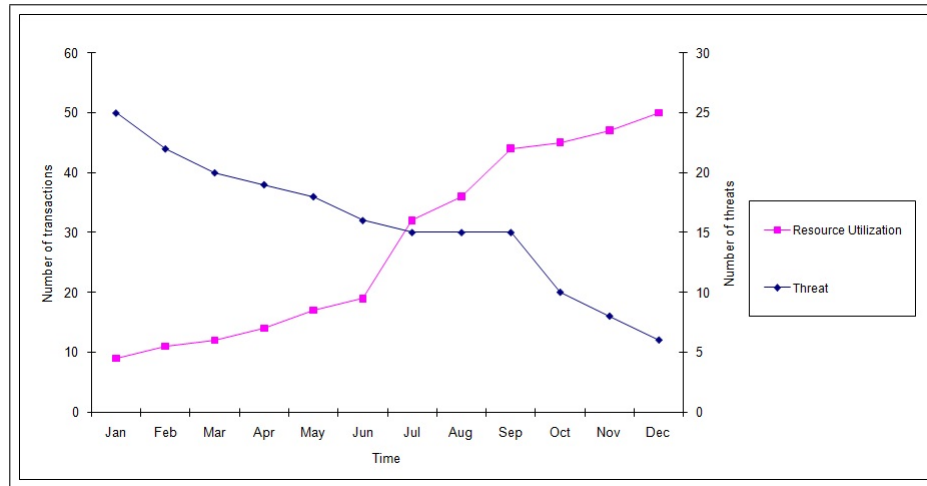


Figure 5.33: Result showing success of Test Case 16: Number of transactions were increased and threats decreased during after deploying TUX-TMS into Grid Computing Environments

- Feature to be tested: Is there any benefit of a trust management in a Grid Computing scenario.
- Pre-condition necessary tasks to do before starting the test execution: Transactions between Service Requester and Service Provider for usage of services provided by TUX-CITMS. Feedback values for trustworthiness and reputation for Service Requester and Service Provider.
- Test procedure: In a Grid Computing scenario, deploy trust management system to automate the process of securing transactions and instilling more confidence in transactions between unknown domains in open distributed grid environments.
- I/P required: Categorization of users on the basis of roles and trust values.
- Post conditions: If the trustworthiness and reputation values are high, it shows the end user is trustworthy and reliable and if the values are low then the end user may be malicious entity.

- O/P: The deployment of a trust management system lead to higher useability of grid in comparison to how grid was available earlier. The results have been deduced while considering various case studies of online transactions. The results are presented in Figure 5.32.
 - Test case pass / fail criteria: Test case is “pass” if honest users are only allowed to use services in Grid Computing scenario and malicious users are checked and prohibited and case is “fail” if honest users are not recognized and malicious users are able to increase scores, use the grid resources for malicious purposes and the grid resources are under security threat.
- (xvi) Test Case 16: In this test case, TUX-CITMS: The Proposed Collaborative Interoperability and Trust Management solution is tested for establishing secure Grids.
- Test case name: Interoperability and Trust Management across Thapar University.
 - Feature to be tested: Is there any benefit of Interoperability and a Trust Management solution to a Grid Computing scenario.
 - Pre-condition necessary tasks to do before starting the test execution: Transactions between Service Requester and Service Provider for usage of services provided by TUX-CITMS.
 - Test procedure: Using TUX-INTERO and TUX-TMS in an integrated manner to secure the grid resources and make all the heterogeneous grid resources available at a single place to an end user.
 - I/P required: Interoperate the resources and apply trust management solution for categorization of users.

Table 5.2: Security Threats on TUX-CITMS

Test Case	Threats
Test Case 1	Authentication, Authorization
Test Case 2	Authentication, Authorization
Test Case 3	Authentication, Authorization
Test Case 4	Certificate Attack, Authentication
Test Case 5	Authorization
Test Case 6	Newcomer Attack
Test Case 7	Sybil Attack, Bad Mouthing, Sleeper Attack
Test Case 8	Sybil Attack, Bad Mouthing, Sleeper Attack
Test Case 9	Sybil Attack, Bad Mouthing, Sleeper Attack
Test Case 10	Sybil Attack, Bad Mouthing, Sleeper Attack
Test Case 11	Authntication, Authorization
Test Case 12	Sybil Attack, Newcomer Attack, Bad Mouthing, Sleeper Attack
Test Case 13	Sybil Attack, Newcomer Attack, Bad Mouthing, Sleeper Attack
Test Case 14	Sybil Attack, Newcomer Attack, Bad Mouthing, Sleeper Attack
Test Case 15	Sybil Attack, Newcomer Attack, Bad Mouthing, Sleeper Attack
Test Case 16	Sybil Attack, Newcomer Attack, Bad Mouthing, Sleeper Attack

- Post conditions: All the resources in a Grid Computing Environment should be available at a single place and should be utilized in a secure manner.
- O/P: The resource utilization was high and threats were less as shown in Figure 5.2.
- Test case pass / fail criteria: Test case is “pass” if all the heterogeneous grid resources are available under single interface and can be used in a secure manner and case is “fail” if there exist malicious use of grid resources and they exist in form of disparate resource islands.

5.1.6 Discussion

The proposed collaborative Interoperability and Trust Management solution is a novel approach to address interoperability and trust management challenges in a Grid Computing environment. The snapshots of database results obtained by aggregating the feedback scores submitted by Service Requester as well as Service Provider in TUX-CITMS have been shown in Figure 5.30 and 5.31 for Service Provider and Service Requester, respectively. The feedback have been taken on an likert scale between 0-1 for each option in Feedback forms and the transactions where the feedback score was submitted is considered for evaluation.

In TUX-CITMS, the users have been clustered on the basis of behavior and ranked according to the trustworthiness and recommendation score in terms of context factor. In this model, higher the trust context factor and higher the reputation of the entity/domain, higher is the rank. The domains with higher ranks in a cluster are allowed access to maximum services. Users are authorized access to resources on the basis of their scores which is a complex task accomplished by TUX-TMS. The users are penalize for malicious activities and false feedback, where their trustworthiness and recommendation scores are discarded for such activities. This decreases their rank in repository which also increases the risk involved in transacting with the domain. Using ranking of domains calculated by considering the parameters as discussed in Chapter 4, TUX-TMS promotes secure transactions. Grid environment which is open becomes secure by knowing the reputation and risk involved in transacting with a domain apriori. Using the systems designed and implemented in the thesis, end users are able to transact with domains having reliable scores. TUX-TMS discourages malicious users from becoming part of grid. It becomes mandatory for users to behave in honest manner to gain scores and services. It secures open environment by encouraging the usage of

trustworthy resources and services by making use of TUX-TMS parameters addressed. From the graphical representation analysis of the results the following conclusion can be drawn:

- As the number of transactions increases, depending on the feedback score given to a Service Requester by a Service Provider, the services allowed to be accessed increases or decreases. Therefore, Service Requesters using services with honest purposes will be allowed to access more services.
- As the number of transactions increases, depending on the feedback score given to a Service Provider by a Service Requester, the services accessed of a Service Provider increases or decreases. Therefore, Service Providers providing trustworthy and secure services will receive more number of requests.
- The Trustworthiness and Reputation score decays with time after a period of inactivity.
- Risk involved in interacting with domains having lower rank is more.
- Trust score inherited is less from domains having more risk in interacting with.

In TUX-CITMS, the interoperability solution, TUX-INTERO is Web Services based solution designed around Windows as well as Linux based systems. It uses Web Services to explore and provide resources to the resource consumer. The advantage that XML Web Services bring to Grid is that a user need not to know in which language the XML Web Services are implemented. The user just needs to know the location of XML Web Service and the methods that the user can call on the Service. Web Services are program components that allows users to build scalable, loosely coupled, platform-independent applications. XML Web Services enable disparate applications

to exchange messages using standard protocols such as HyperText Transfer Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL). XML Web Services use XML based messaging to send and receive data, which enables heterogeneous applications to interoperate with each other. TUX-INTERO is designed around Web Services which is already in the process of standardization as it is the key to Interoperability. By incorporating Windows based systems into Grid will enhance Grid usage and adoption.

In TUX-CITMS, the trust management solution, TUX-TMS is flexible Reputation based trust management system. Trust and reputation systems represent a significant trend in decision support for Internet mediated service provision. The basic idea is to let parties rate each other, for example after the completion of a transaction and use the aggregated ratings about a given party to derive a trust or reputation score, which can assist other parties in deciding whether or not to transact with that party in the future. The existing trust management solutions are Policy based or Reputation based. However, in case of existing trust management solutions in Grid, only policy based solutions are used, which incurs heavy cost on the system and users. TUX-TMS is a flexible Reputation based Trust Management system which combines the merit of policy proof and reputation evaluation. TUX-TMS also considers parameters such as Intrusion Detection System (IDS) and Audit Trails to promote secure transactions in Grid.

In the concluding remarks, results of TUX-CITMS show that the proposed Interoperability and Trust Management can achieve high transaction success rate by providing secure services. TUX-TMS provided flexible and robust security related functionality by addressing identity trust using policies to authorize entities. Therefore, the use of collaborative interoperability and trust management solutions establishes secure Grids.

The thesis clearly depicts that a collaborative interoperability and trust management solution will lay the foundation for secure Grids. Until recently, focus in Interoperability and Trust Management has been on either providing interoperability or trust management solutions in a Grid. However, TUX-CITMS addresses both Interoperability and Trust Management challenge together in a Grid. Binding trust information and providing a threshold for accessing Grid services encourages minimal chances of malicious use or security attacks on resources.

The next section discusses TUX-INTEROP, the other proposed interoperability solution that, is a portal for secure Interoperation of grids.

5.2 TUX-INTEROP: A Portal for secure interoperation of grids

TUX-INTEROP Portal a testbed has been created which mainly consists of nodes on different middleware. It has been deployed at Thapar University Campus and mainly managed from the Centre of Excellence in Grid Computing (CoEGC). TUX-INTEROP provides an interactive and easy to use interface for users to facilitate access to Grid resources. The details of the experimentation are presented as a case study conducted on Thapar University Campus Grid where the resources were used from Centre of Excellence in Grid Computing, ME (Software Lab) and CCMX-I lab.

Basically there are three groups of stakeholder in this setup: the Service Providers/Resources Owners, the Service Requesters/Resource Consumers and Administrator.

Service Providers/Resource Owners: The resources from the three different labs were considered to be belonging to three different resource owners.

Service Requesters/Resource Consumers: Resource Consumers are students and faculty of Thapar University who have utilized the resources through TUX-INTEROP. A Grid user account is needed to access TUX-INTEROP.

Administrator: Administrator controls and monitors the activities of Service Provider and Service Requester to utilize resources available through TUX-INTEROP. These stake holders are considered essential in a grid setup. Then, there are evaluation parameters necessary to evaluate the useability of TUX-INTEROP to address Interoperability challenge as discussed in the next section.

5.2.1 Evaluation Parameters

The parameter considered for evaluation using TUX-INTEROP is interoperability in grids which is at the level of:

- Security: The Portal should be able to support multiple resources from different platforms using a single sign-on facility.
- Job Management: The Portal should be able to run jobs on various resources irrespective of the platforms they are hosted on.
- File Management: The Portal should be able to provide data as and when required at a particular place for execution purposes.
- Resource Management: The Portal should keep the intricacies of grid hidden from the user.

These parameters are deemed essential for an Interoperability system or solution in a Grid. The next section presents experimental details of TUX-INTEROP.

5.2.2 Experimental Details

TUX-INTEROP has been designed in such a way that it provides access of heterogeneous resources to the resource consumer independent of the platform, the resource is hosted in. TUX-INTEROP is easily deployable and installable. A simple script is provided to deploy the portlets into a portlet container running in a web server. Each information service is queried at a specified poll interval, so communication overhead is controlled by varying the length of the interval. The Resource Registry Portlet in TUX-INTEROP is resource information. A number of functions can be applied to it as mentioned below:

Customizing the Portal: The TUX-INTEROP resource registry portlet allows administration of grid resources and general presentation details of the portal. Here, resources can be added and removed and existing resources; details can be modified.

Adding a new resource: Adding a new resource requires simply entering the host-name and then submitting the form. This will add the new resource to the list of existing resources and select it for editing so that its details can be entered in resource registry portlet.

Editing resource details: To change the properties of a resource, select the desired resource. This will present the properties of that resource in a form allowing them to be modified. The hostname and type must be set correctly for the resources information to be successfully queried by the resource manager.

Deleting a resource: Resources that are no longer part of your Grid can be deleted from the system while selected for editing. This will permanently remove the resource and all other information that was gathered.

Browsing the Portal: Once customization is complete the portal can be made available to users to make use by submitting jobs to the resources. The job management

portlet requires an end user to configure the job according to his requirements and fill the necessary parameters. The file management portlet is responsible to provide files to upload and download to the end users. The resources can be monitored for job execution.

5.2.3 Testing Details

Different test cases were taken for testing the proposed solutions. The test cases and testing results are discussed in this section.

(i) Test Case 1: In this test case, TUX-INTEROP: The Proposed Grid Interoperability Portal solution is tested for security interoperability for running job across various heterogeneous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for security.
- Feature to be tested: Is there any benefit of security in a grid portal.
- Test environment: required h/w and s/w to execute these test cases: Heterogeneous resources hosted on various middleware such as Globus Toolkit (GT) and UNICORE 6. Servlet container Apache Tomcat 5.5.26 and Portal container GridSphere 3.1 should be running successfully. MyProxy server should be installed.
- Pre-condition necessary tasks to do before starting the test execution: Servlet container and portlet container should be running and resources should be configured properly with the location the Portal server is deployed on. MyProxy server should be running.

- Test procedure: Configure the resources for discovery purposes. When a user logs in and submits MyProxy credential should be able to run jobs on the resources available.
- I/P required: Varied security authentication procedures.
- Post conditions: Portal should be able to host multiple resources from other security policies.
- O/P: Number of resources available were high as compared to resources available without authentication Interoperability. Figure 5.34 and Figure 5.35 shows the login screen of TUX-INTEROP.
- Test case pass / fail criteria: Test case is “pass” if portal is able to discover and present resources having multiple security authentication policies and case is “fail” if portal is not able to discover the resources from multiple administrative domains.

The Test environment and Pre-Condition remains same for Test Cases 2-4 described below.

(ii) Test Case 2: In this test case, TUX-INTEROP: The Proposed Grid Interoperability Portal solution is tested for interoperability for managing data through file management across various heterogenous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for file management
- Feature to be tested: Is there any benefit of file management in a grid portal.
- Test procedure: In the file management portlet, an end user should be able to upload and download the files necessary for job execution at the resource

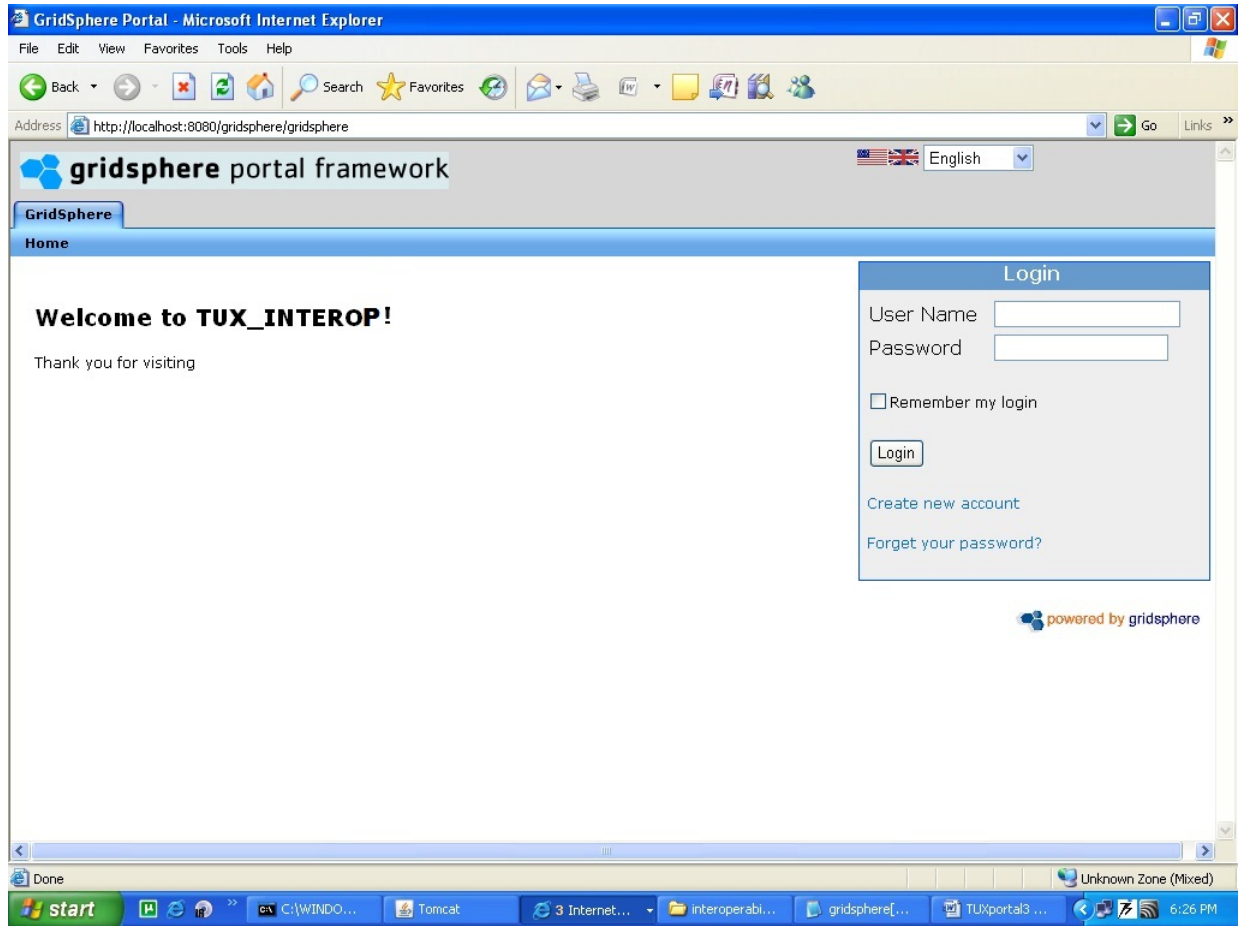


Figure 5.34: Result showing success of Test Case 1:TUX-INTEROP Login Screen designed in Gridsphere

where it is required.

- I/P required: Varied files extensions should be supported.
- Post conditions: Portal should be able to download output files and upload input files required for job execution. Files should be easily manageable and results should be available to the end user.
- O/P: Figure 5.36 successfully shows the files uploaded and can be easily downloaded as and when required.
- Test case pass / fail criteria: Test case is "pass" if portal is able to upload and download files as required for job execution on grid and case is "fail" if

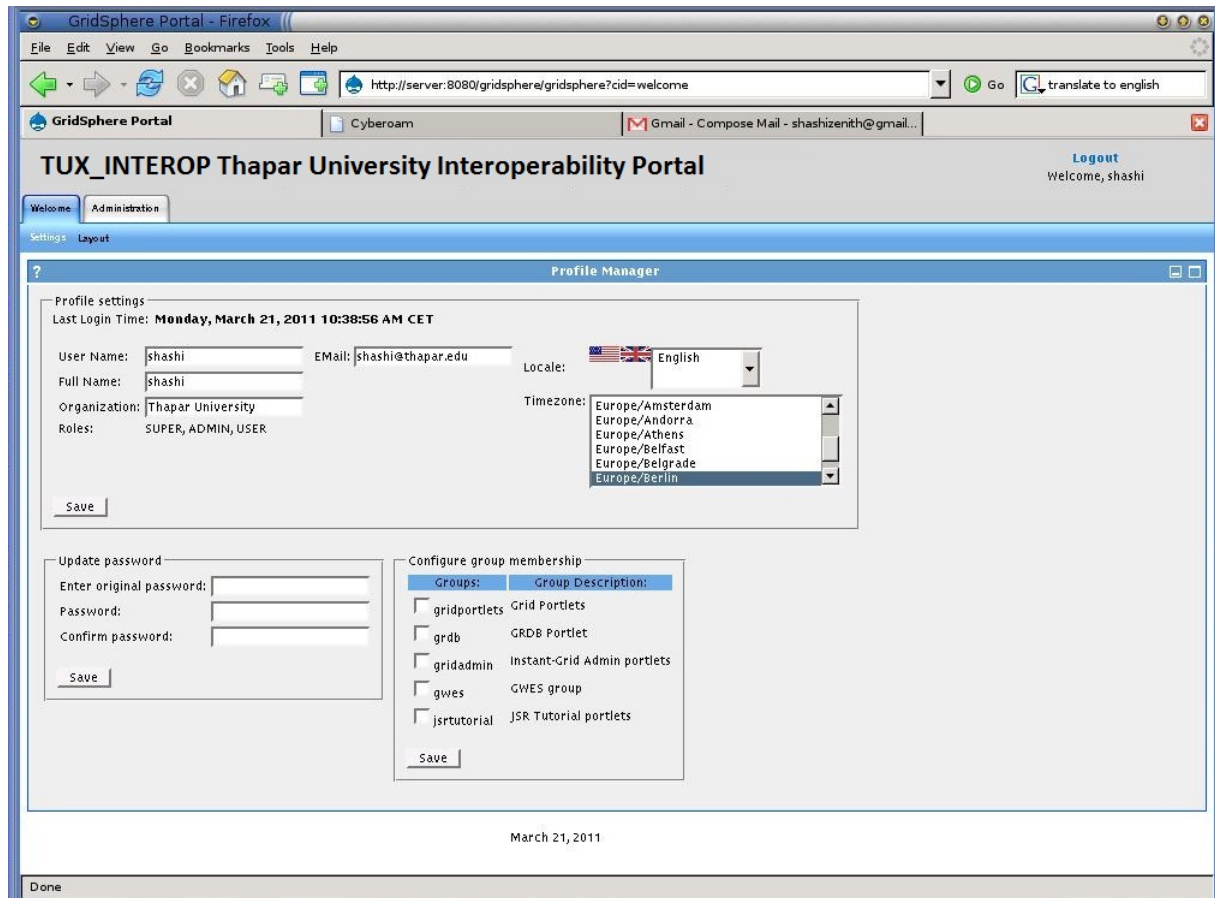


Figure 5.35: TUX-INTEROP Portlets

portal is not able to upload and download files as required for job execution on grid.

(iii) Test Case 3: In this test case, TUX-INTEROP: The Proposed Grid Interoperability Portal solution is tested for interoperability for providing access to heterogeneous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for resource management

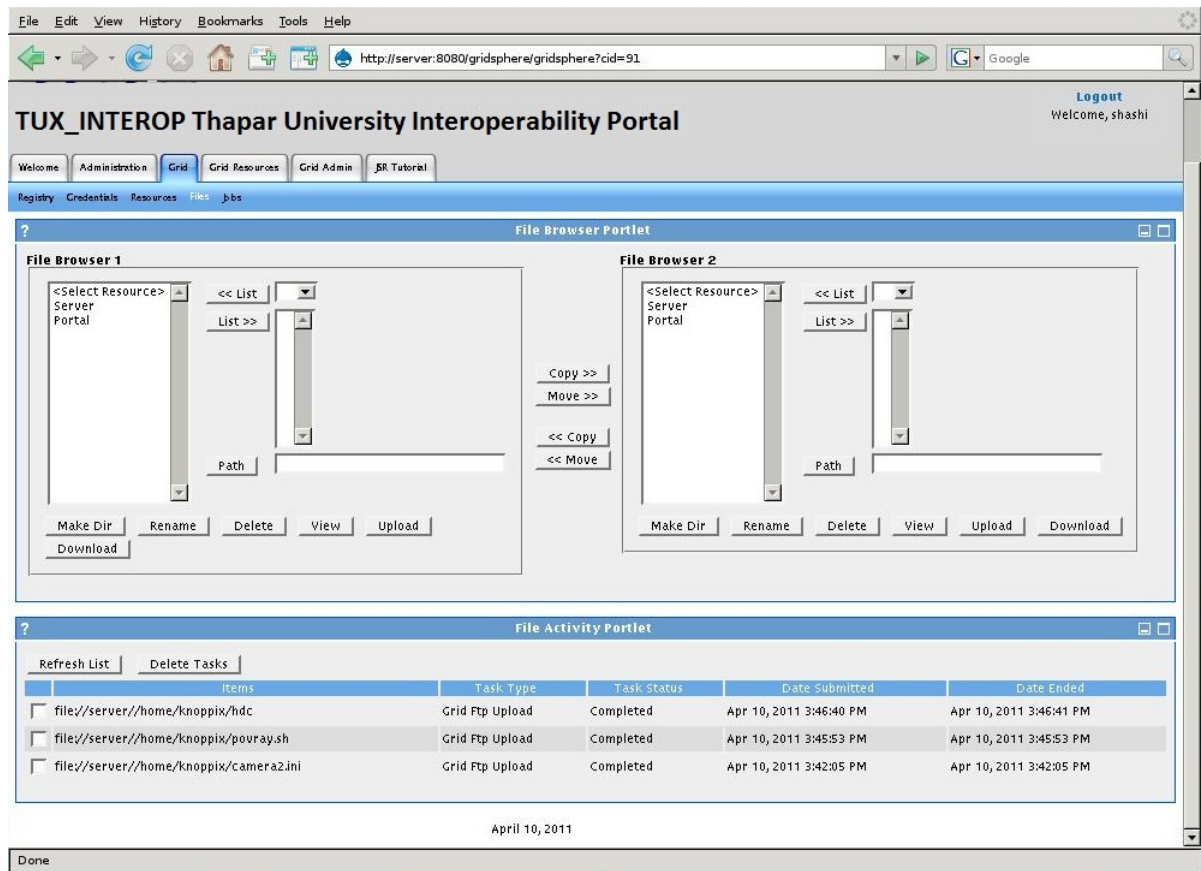


Figure 5.36: Result showing success of Test Case 2: File Management Portlet of TUX-INTEROP

- Feature to be tested: Is there any benefit of resource management in a grid portal.
- Test procedure: In the resource management portlet, an end user should be able to configure various heterogeneous resources.
- I/P required: Varied resource descriptions.
- Post conditions: Portal should be able to discover any kind of resource configured which may be hosted on multiple platforms. Number of resources available were high as compared to resources available without resource management in portal.

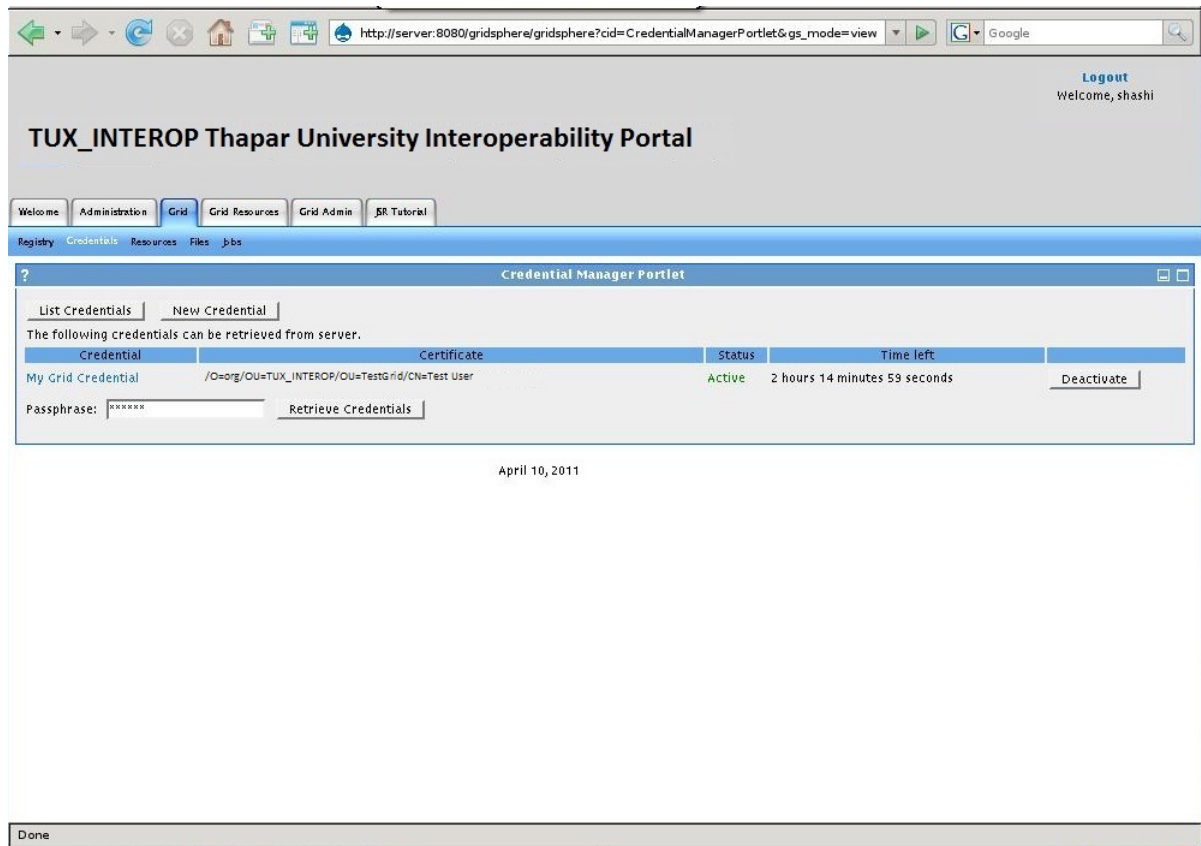


Figure 5.37: Credential Management Portlet of TUX-INTEROP

- O/P: Figure 5.38 successfully exhibit the resources available.
 - Test case pass / fail criteria: Test case is "pass" if portal is able to run a discover resource requested and find results and case is "fail" if portal is not able to run the job on various resources available through the portal.
- (iv) Test Case 4: In this test case, TUX-INTEROP: The Proposed Grid Interoperability Portal solution is tested for job management interoperability for running job across various heterogenous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.
- Test case name: Interoperability across Thapar University for job management.



Figure 5.38: Result showing success of Test Case 3: Resources in TUX-INTEROP

- Feature to be tested: Is there any benefit of job management in a grid portal.
- Test procedure: In the job management portlet, an end user should be able to input job parameters and run the job on the resource requested.
- I/P required: Varied job requirements.
- Post conditions: Number of jobs executed were high as compared to job executed without job management in portal. Portal should be able to run any kind of job on multiple resources available.
- O/P: Figure 5.37 shows the credential to be uploaded to enable the deployment of jobs on resources. Figure 5.39 shows the job management portlet and the job can be submitted successfully from here. Figure 5.38 and Figure

5.40 monitors the status of resource on which job is submitted.

- Test case pass / fail criteria: Test case is "pass" if portal is able to run a job on the resource requested and find results and case is "fail" if portal is not able to run the job on various resources available through the portal.

The screenshot shows a web application window titled "Job Submission Portlet". At the top, there are tabs: "Welcome", "Administration", "Grid", "Grid Resources", "Grid Admin", and "JR Tutorial". Below these is a sub-menu: "Registry", "Credentials", "Resources", "Files", and "Jobs". The main content area is titled "Step 1. Specify the application you would like to execute." and contains the following fields:

- Description:** A text input field.
- Executable:** A text input field containing "file://server//home/shashi" and a "Browse" button.
- Type of executable:** A dropdown menu with options "multiple", "single", "mpi", and "condor".
- Directory:** A text input field and a "Browse" button.
- Stdout:** A text input field and a "Browse" button.
- Stderr:** A text input field and a "Browse" button.
- Arguments:** A large text area.
- Environment:** A large text area.
- Stage-In Files:** A large text area.
- Stage-Out Files:** A large text area.

At the bottom of the window, there is a "Done" button.

Figure 5.39: Result showing success of Test Case 4: Job Management Portlet of TUX-INTEROP

5.2.4 Discussion

TUX-INTEROP provides a simple mechanism to view relevant resource information and access heterogeneous resources located in various domains connected via local resource management systems and middleware. There are currently a number of portals designed to discover and access resources, however, it is difficult to integrate diverse

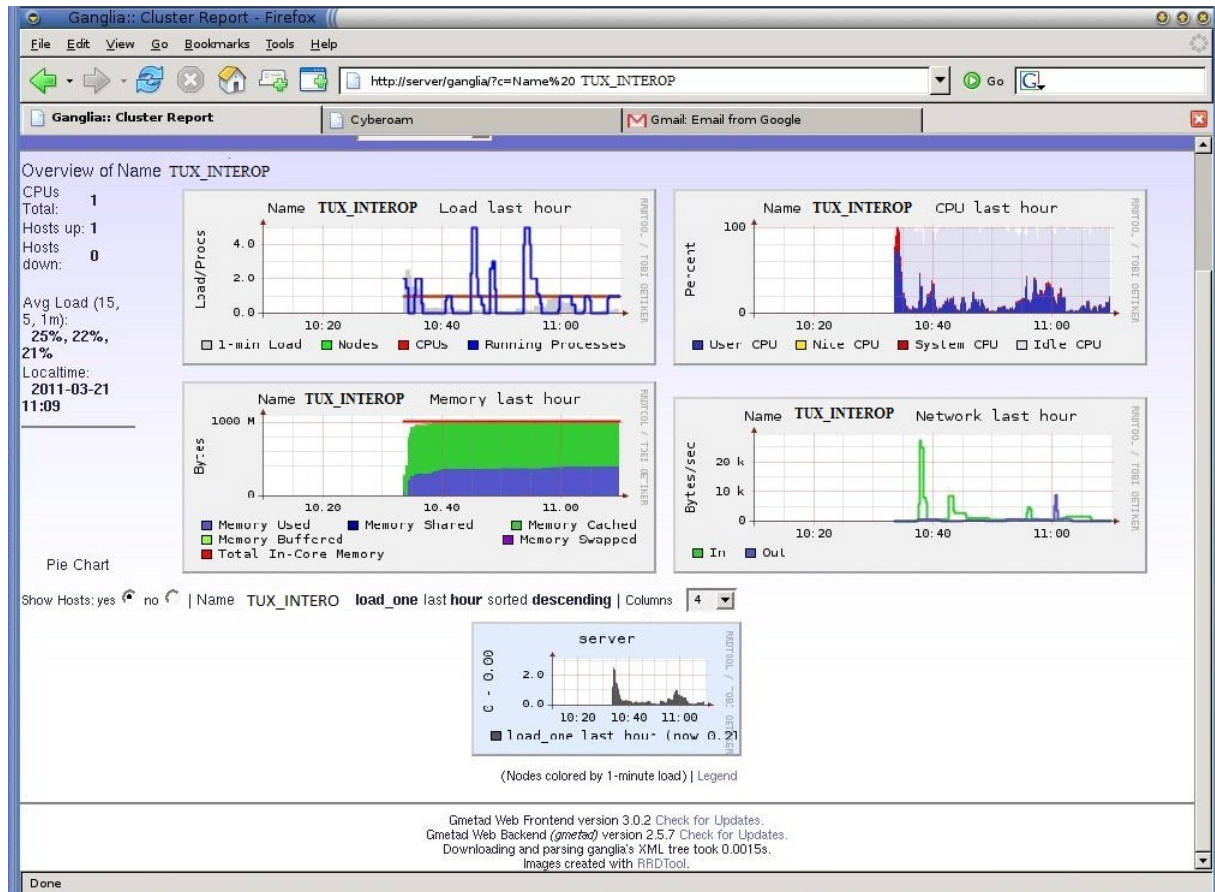


Figure 5.40: Monitoring a server in TUX-INTEROP

information into a single interface or customize and apply them in other scenarios or Grids. TUX-INTEROP provides a highly flexible solution that allows integration with current and future information sources, as well as providing an easy mechanism for customizing which information is to be displayed and how it should be presented. Essentially, TUX-INTEROP can be applied in many situations and need not be restricted to any specific middleware or Grid project or even to Grids in general. However, it becomes difficult to extend the portal for further usage due to lot of effort required in programming. It motivated the development of portal based through Vine Toolkit as discussed in next section.

5.3 PIE

PIE is an effort to design and develop pluggable, interoperable and extensible portals which entail less programming effort from scientists and developers. PIE has been designed with the aim of allowing end users to easily run jobs on grid and allow pluggable portlets to be embedded into portal as per use. Portal based solutions help users to achieve interoperability and collaborate disparate grids. It achieves the original vision of Grid which is aimed at single massive infrastructure. Here, any user is allowed to submit job instead of getting a user trained in special type of software or technique. The next section discusses parameters for evaluation of PIE.

5.3.1 Evaluation Parameters

The parameter considered for evaluation using PIE is interoperability in grids which is at the level of:

- Job Submission: The Portal should be able to run jobs on various resources irrespective of the platforms they are hosted on.
- Resource Description: The Portal should keep the intricacies of grid hidden from the user.
- Security: The Portal should be able to support multiple resources from different platforms using a single sign-on facility.
- Pluggability and Extensibility: The Portal should be able to easily plug and extend other portlets.

These parameters are deemed essential for an Interoperability system or solution.

5.3.2 Experimental Details

PIE aims at allowing easy deployment and usage of grid resources. The experimental details are elaborated in form of a use case scenario here:

Managing certificates: After defining the grid environments the user should upload his grid certificates belonging to the connected grids into the MyProxy server that works with strong interaction with the portal. Of course, if several grids accept the same certificate it is enough to upload their common certificate. The Certificate Manager as shown in Figure 5.41 takes care of certificates required for the user to run the jobs. The Certificate Manager is installed and used separately to store different credentials of the user.

Login into PIE: The login details are available to user through mailing the details to the PIE administrator which checks for credential authenticity. The users are authorized to access resources based on distinguished name (DN) mapping of their credentials provided as shown in Figure 5.42. The user accounts details are e-mailed to the user. The user can use the account details to login in PIE Login screen as shown in Figure 5.43. New credential can be managed using Credentials Manager Portlet as shown in Figure 5.44.

Running a Job: Once the resources are configured as show in Figure 5.45 of Registry Portlet they can be simply used, the next step is to upload the files to run the job. The File management Portlet contains the files required by the user as shown in Figure 5.46. These files can be uploaded and downloaded at a specific location as and when required. The Job Management portlets in Figure 5.47 and Figure 5.48 enables an end user to submit a job in a standardized format Job Service Description Language (JSDL) to the resource selected.

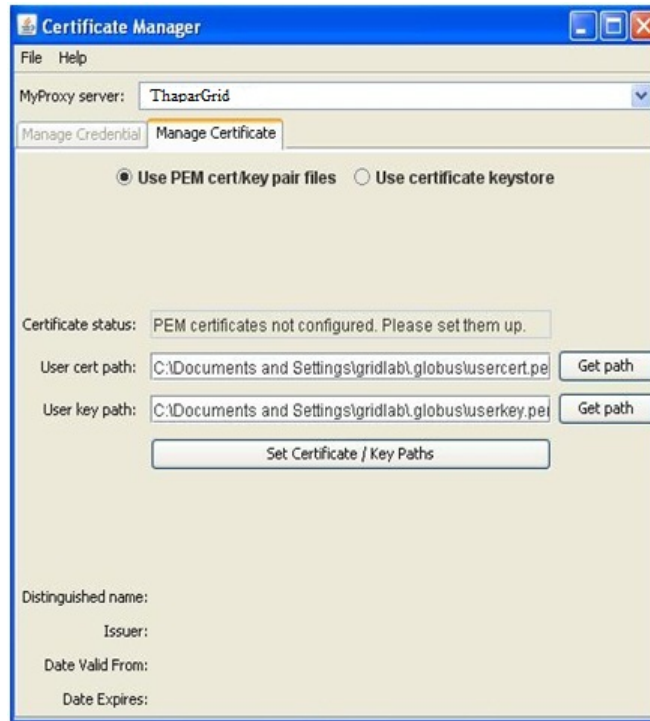


Figure 5.41: Certificate Manager

5.3.3 Testing Details

The test cases considered for testing PIE are as follows:

Different test cases were taken for testing the proposed solutions. The test cases and testing results are discussed in this section.

- (i) Test Case 1: In this test case, PIE: The Proposed Pluggable, Interoperable and Extensible Grid Portal is tested for interoperability for providing single sign on access to users of various heterogenous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for security.
- Feature to be tested: Is there any benefit of security in a grid portal.

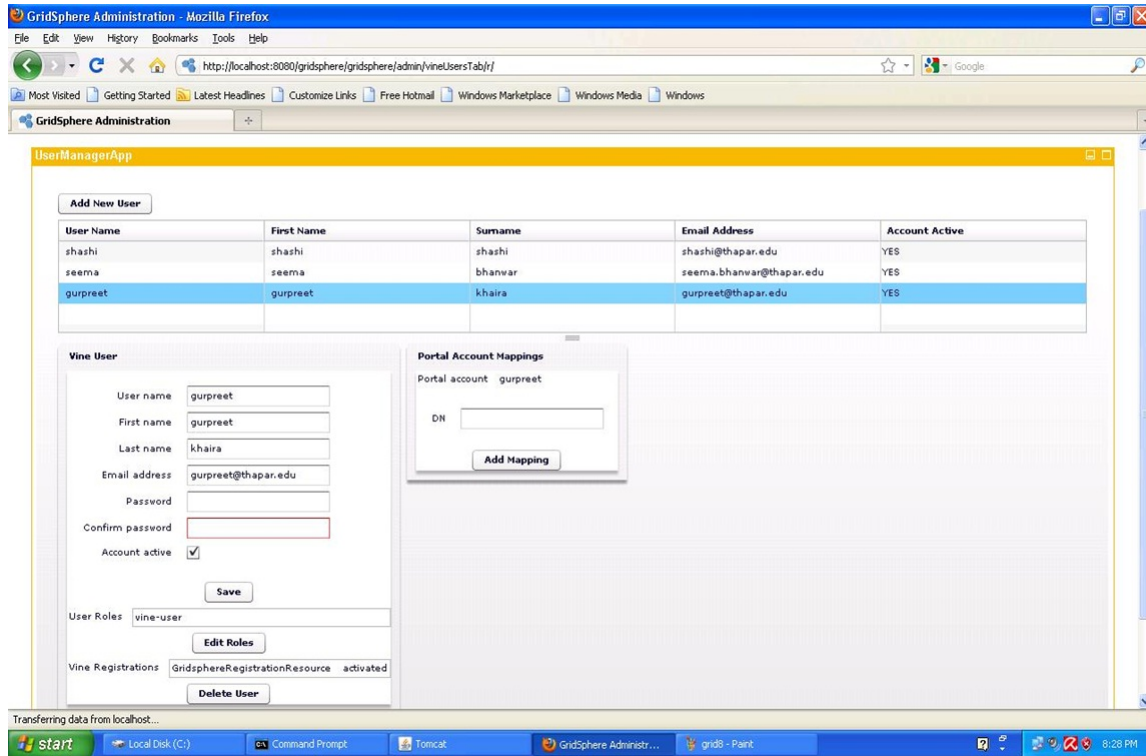


Figure 5.42: Mapping Roles to the PIE users

- Test environment: required h/w and s/w to execute these test cases: Heterogeneous resources hosted on various middleware such as Globus Toolkit (GT) and UNICORE 6. Servlet container Apache Tomcat 5.5.26 and Portal container GridSphere 3.1 with Vine toolkit 1.2 should be running successfully. MyProxy server should be installed and Certificate Manager should be running.
- Pre-condition necessary tasks to do before starting the test execution: Servlet container and portlet container should be running and resources should be configured properly with the location the Portal server is deployed on. MyProxy server should be running.
- Test procedure: Configuring the resources using domain.xml file for multiple security policies for authentication. When a user logs in should be able to

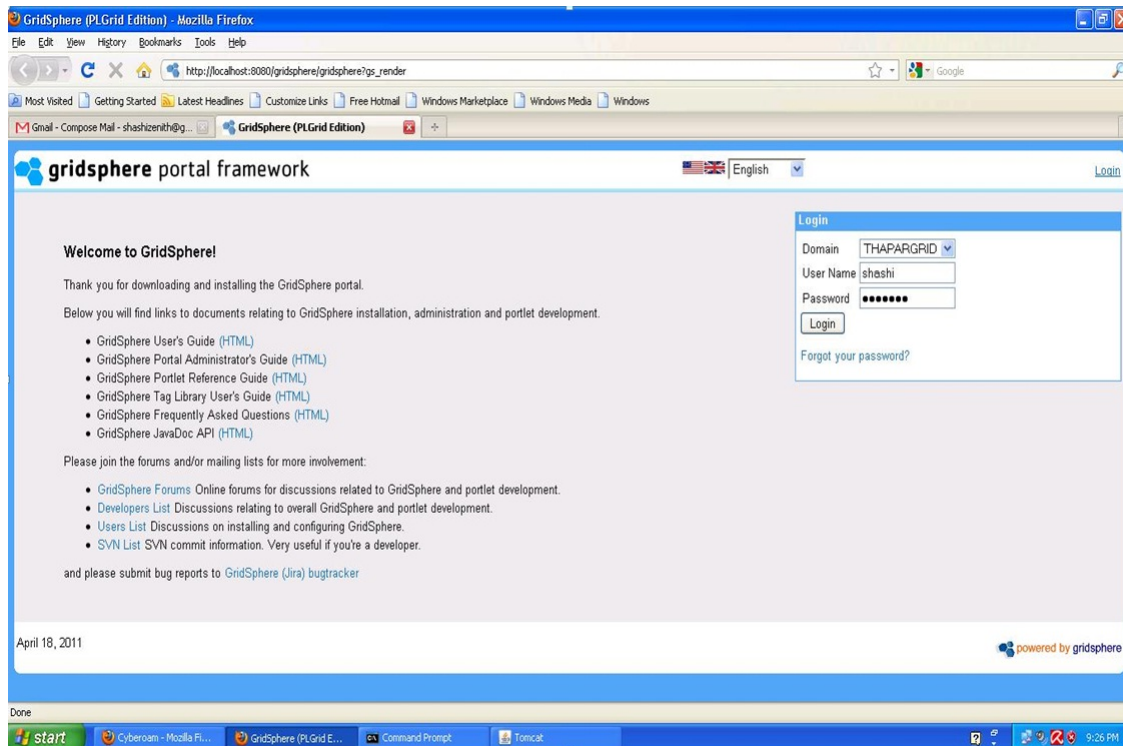


Figure 5.43: Login Screen of PIE

locate resources to run jobs on.

- I/P required: Varied security authentication procedures.
- Post conditions: Portal should be able to host multiple resources from other security policies.
- O/P: Number of resources available were high as compared to resources available without authentication Interoperability. Figure 5.44 shows the credentials can be used from multiple domains.
- Test case pass / fail criteria: Test case is “pass” if portal is able to discover and present resources having multiple security authentication policies and case is “fail” if portal is not able to discover the resources from multiple administrative domains.

The screenshot displays the CredentialManagerApp interface. At the top, there are tabs for 'Credentials', 'Files', 'Jobs', and 'Resources'. Below the tabs, the application title 'CredentialManagerApp' is shown. A toolbar contains buttons for 'New credential', 'Edit credential', 'Deactivate credential', and 'Delete credential'. A table with four columns (DN, Label, Status, Time remaining) is visible, currently empty. Below the table, a 'New credential' form is open, containing fields for Label (Myproxy credential), User name (research.gridlab.com), Name (research), Lifetime (8100), and Passphrase (masked with asterisks). 'Save credential' and 'Cancel' buttons are at the bottom of the form. A status bar at the very bottom indicates 'transferring data from localhost...'.

Figure 5.44: Result showing success of Test Case 1: Credential Management Portlet of PIE

The Test environment and Pre-Condition remains same for Test Case 2-5 described below.

(ii) Test Case 2: In this test case, PIE: The Proposed Pluggable, Interoperable and Extensible Grid Portal is tested for interoperability for running job across various heterogenous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for job management.
- Feature to be tested: Is there any benefit of job management in a grid portal.
- Test procedure: In the job management portlet, an end user should be able to input job parameters and run the job on the resource requested.

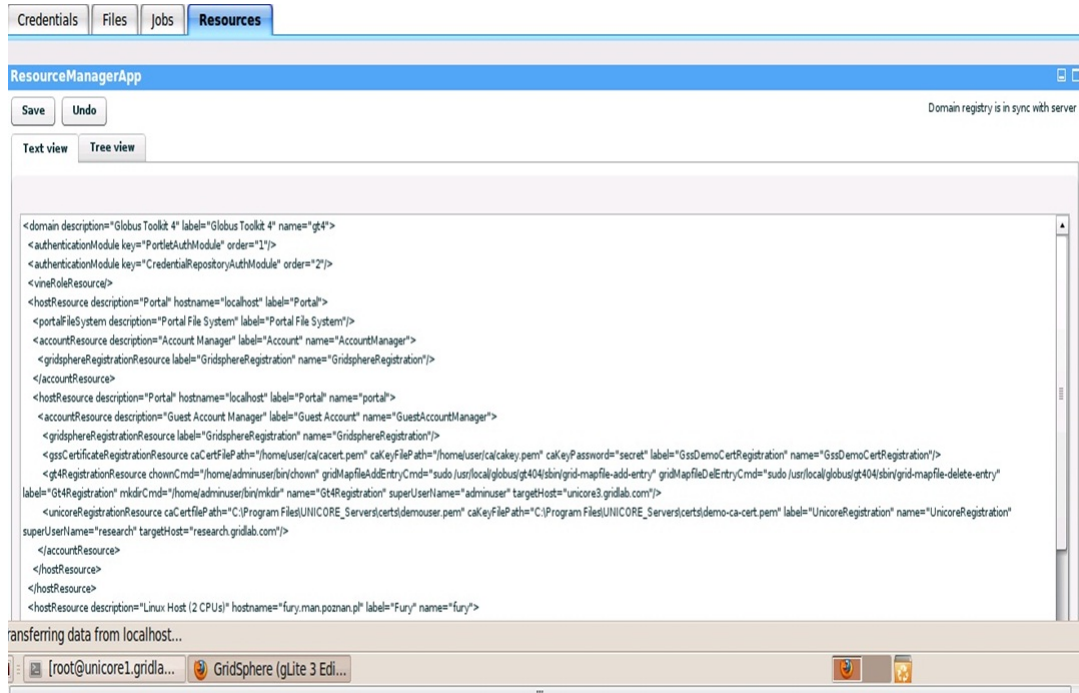


Figure 5.45: Result showing success of Test Case 4: Registry Management Portlet of PIE

- I/P required: Varied job requirements.
 - Post conditions: Portal should be able to run any kind of job on multiple resources available.
 - O/P: Number of jobs executed were high as compared to jobs executed without job management in portal. Figure 5.47 and 5.48 shows the job management portlet of PIE.
 - Test case pass / fail criteria: Test case is “pass” if portal is able to run a job on the resource requested and find results and case is “fail” if portal is not able to run the job on various resources available through the portal.
- (iii) Test Case 3: In this test case, PIE: The Proposed Pluggable, Interoperable and Extensible Grid Portal is tested for interoperability for managing data through file management across various heterogenous resources distributed across various

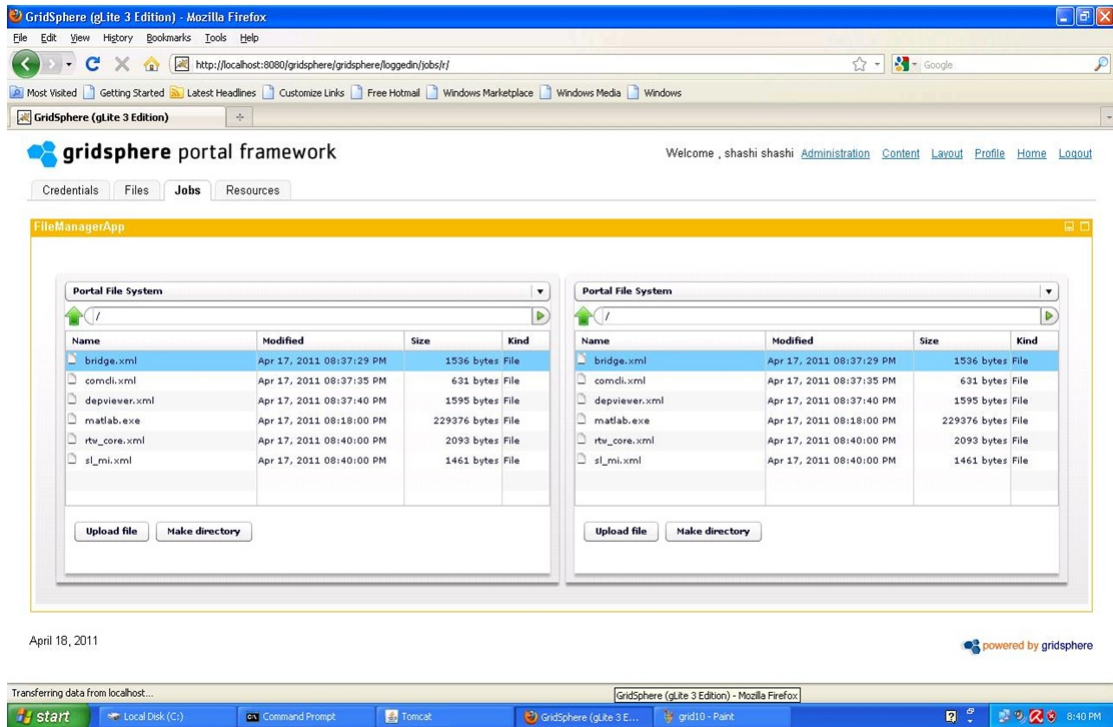


Figure 5.46: Result showing success of Test Case 3: File Management Portlet of PIE

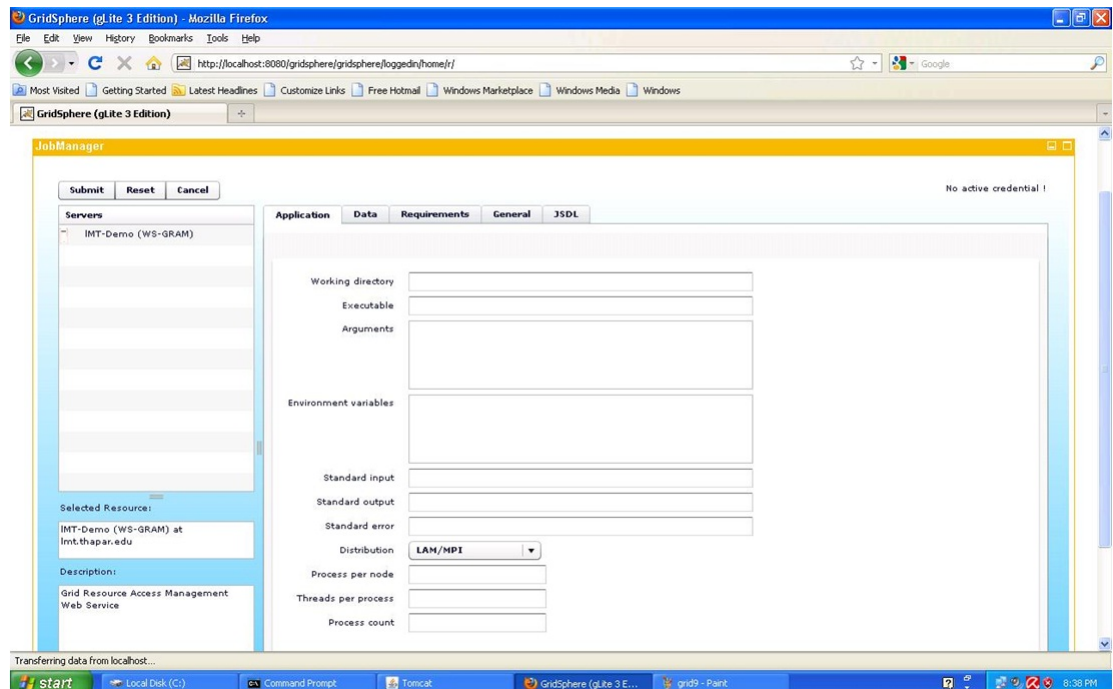


Figure 5.47: Result showing success of Test Case 2: Job Management Portlet of PIE

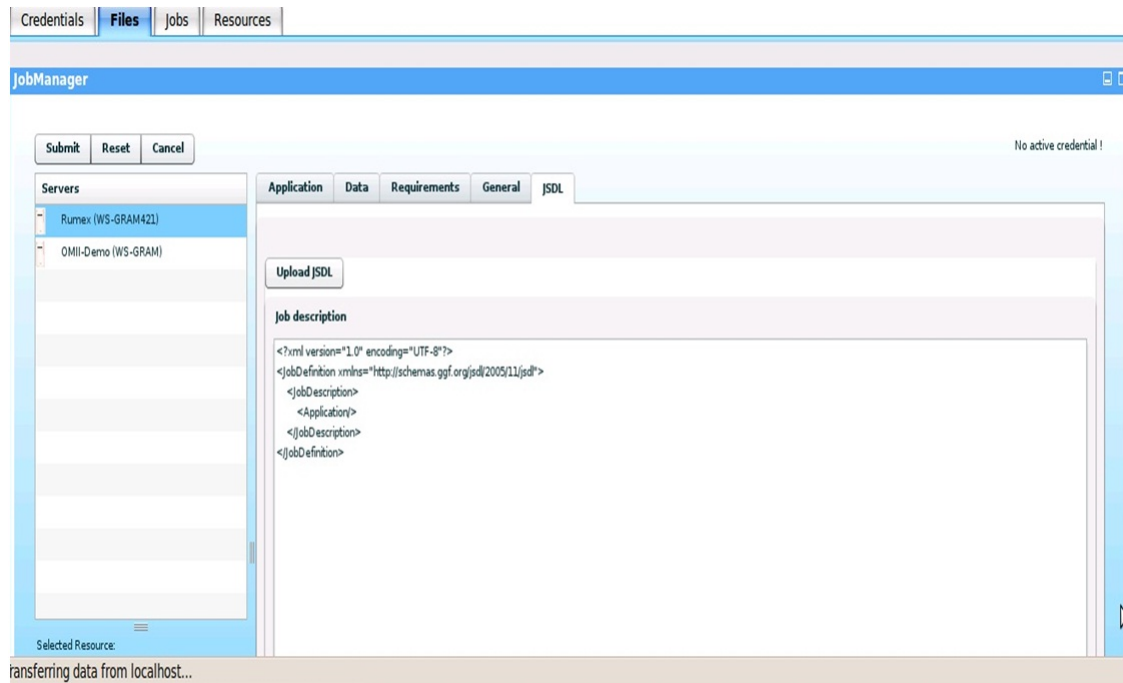


Figure 5.48: Writing a job using Job Service Description Language (JSDL)

labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for file management
- Feature to be tested: Is there any benefit of file management in a grid portal.
- Test procedure: In the file management portlet, an end user should be able to upload and download the files necessary for job execution at the resource where it is required.
- I/P required: Varied files extensions should be supported.
- Post conditions: Portal should be able to download output files and upload input files required for job execution.
- O/P: Files should be easily manageable and results should be available to the end user. Figure 5.46 successfully shows that the files are uploaded and

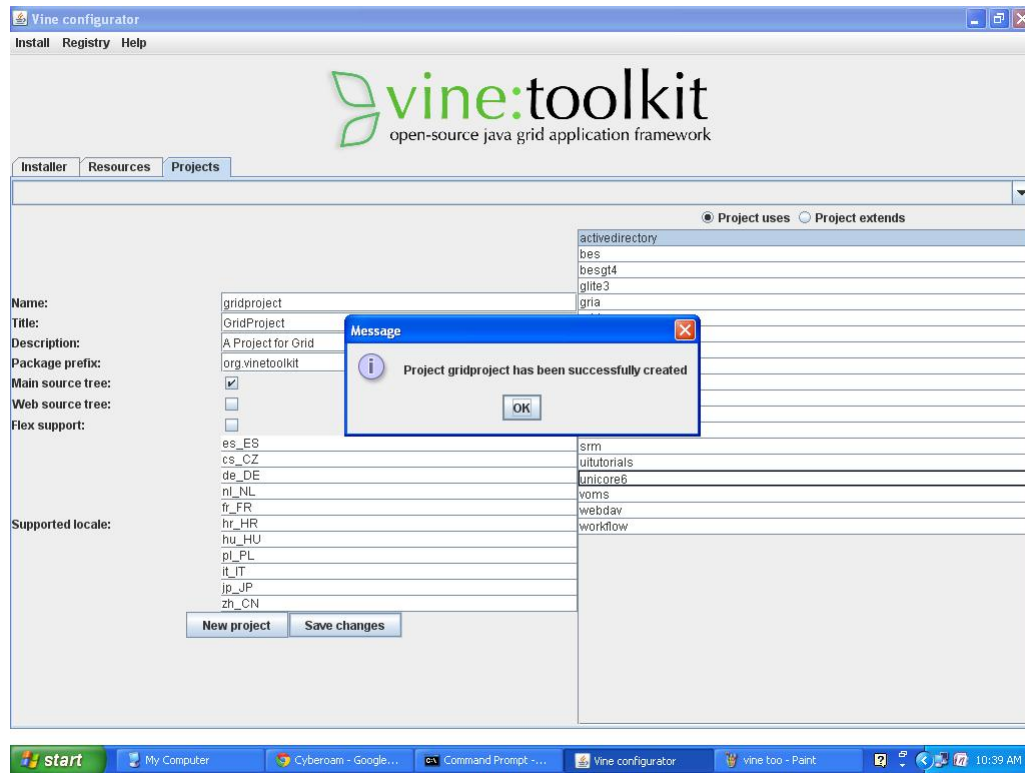


Figure 5.49: Result showing success of Test Case 5: Creating and embedding new portlets using a Project

downloaded easily from PIE.

- Test case pass / fail criteria: Test case is "pass" if portal is able to upload and download files as required for job execution on grid and case is "fail" if portal is not able to upload and download files as required for job execution on grid.
- (iv) Test Case 4: In this test case, PIE: The Proposed Pluggable, Interoperable and Extensible Grid Portal is tested for interoperability for discovering various heterogeneous resources distributed across various labs in Computer Science and Engineering Department of Thapar University.

- Test case name: Interoperability across Thapar University for resource management.
 - Feature to be tested: Is there any benefit of resource management in a grid portal.
 - Test procedure: In the resource registry portlet, an end user should be able to input details of resources and configure them for usage.
 - I/P required: Varied resource descriptions.
 - Post conditions: Portal should be able to discover any kind of resource configured which may be hosted on multiple platforms.
 - O/P: Number of resources available were high as compared to resources available without resource management in portal as shown in Figure 5.45. Figure 5.47 shows the resources available belonging to various middleware in PIE.
 - Test case pass / fail criteria: Test case is "pass" if portal is able to configure a resource and use it and case is "fail" if portal is not able to configure the resource or use it.
- (v) Test Case 5: In this test case, PIE: The Proposed Pluggable, Interoperable and Extensible Grid Portal is tested for pluggability and extensibility of various portlets.
- Test case name: Pluggability and Extensibility.
 - Feature to be tested: Is there any benefit of pluggability and extensibility in a grid portal.
 - Test procedure: In PIE, an end user should be able to plug other portlets as per functionality.

- I/P required: Varied requirements as per functionality.
- Post conditions: Portal should be able to plug and extend the portlets easily.
- O/P: Ease of use to operate a portal and extensibility increases with plug-gability and extensibility in a portal. Figure 5.49 shows a gridproject added successfully with new portlets as per functionality.
- Test case pass / fail criteria: Test case is “pass” if portal is able to plug other portlets easily and case is “fail” if portal is not able to plug portlets.

5.3.4 Discussion

PIE Portal is a set of Vine grid portlets running in Gridsphere portal framework. The interoperability is supported by several functions in PIE portal. A Grid is typically identified by its information system and hence when PIE portal is installed, the portal administrator can connect the portal to several grid resources. PIE provides a uniform interface to high performance computing infrastructures by mapping and configuring heterogeneous resources using Resource Registry portlet's domain.xml. The Resource information can be configured directly in Resource Registry Portlet or may be specified using domain.xml file. The domain.xml file has been configured for the resources available at Centre of Excellence in Grid Computing at Thapar University. PIE provides interoperability at the level of single sign on. PIE also deals with security issues. It can solve different security issues while accessing different grid services. A significant interoperability problem results from the differences in the security models of systems such as UNICORE and Globus-based Grids (authentication based on GSI). The main difference between these systems is that a basic version of UNICORE and GLOBUS does not support dynamic delegation while GSI-compliant systems do. Currently, to overcome this problem, PIE authenticates on behalf of end-users. In case of

UNICORE, PIE supports GSI-enabled extension for UNICORE gateway what enables to authenticate users by using proxy certificates. Account creation allows automated, user-friendly user registration at the portal and to underlying Grid middleware and third-party services used through the portal (registration modules are used to register new user accounts in external services). User authentication allows the user to login into the portal and be authenticated. Authorization allows the administrator to access third-party authorization systems to change user permissions. Common problem to solve while using different middleware infrastructures is user management and account provision. PIE is able to register users in different middleware infrastructures like Unicore 6 or Globus GT4.

PIE addresses security issues which always occur while accessing different middleware stacks and solved them by using Vine registration, authorization modules and proxy certificates support. Role management mechanism allows administrators and end users to easily manage permissions.

5.4 Conclusions

In this Chapter deployment details, experimental setup and testing results of the proposed Interoperability and Trust Management solutions have been demonstrated. TUX-INTERO and TUX-TMS have been integrated, deployed and tested on TUX-CITMS. TUX-CITMS, a Grid Testbed has been specifically developed for testing purposes. This testbed spans across whole Thapar University Campus including entire academic area, some part of residential areas, utility area like library, cyber surfing labs, cafeteria etc. Testing results have been presented in the form of sixteen test cases. Every test case defines a testing criteria, test inputs, expected outputs and

finally validates the outcome to be correct/ as expected. TUX-INTEROP and PIE have been deployed and tested further on Thapar University (TU) Campus Grid. TU Campus Grid spans across Centre of Excellence in Grid Computing (CoEGC) lab, ME (Software Lab) and CCMX-I lab of Computer Science and Engineering Department for resources. The users of TU Campus Grid are faculty members and students of Thapar University. Testing Results of TUX-INTEROP and PIE have been demonstrated using four and five test cases, respectively. Every test case defines a testing criteria, method, pre and post conditions and finally validates the outcome. The results successfully exhibit interoperability between resources on multiple platforms and middleware. The testing results collectively demonstrate that the proposed solutions are working and can be effectively used to address interoperability and trust management challenges to establish secure Grids.

The next chapter concludes the thesis and also shows a future direction.

Chapter 6

Conclusions and Future Directions

This is the concluding chapter of the thesis. The chapter begins with a summary of the contributions made in this thesis. Next, objectives that have been achieved in this thesis are mentioned. Finally, possible extensions of the present work to achieve some other goals in the current domain as well as other domains are discussed. Several possible directions of future research are also discussed.

6.1 Summary of Contributions

The thesis, “Interoperability and Trust Management Solutions for establishing secure Grids” addresses Interoperability and Trust Management challenges in Grid Computing. As an outcome, three Interoperability Solutions: Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE) have been proposed and developed. As Trust Management solution: Thapar University Extensible Trust management System (TUX-TMS) has been proposed and developed. The design, development and testing of the proposed Interoperability and trust man-

agement solutions has been presented and successfully demonstrated in this thesis. It focusses on the issues related to security in a Grid for wide adoption and realizing the vision of Grid into a single and reliable massive infrastructure.

- (i) In chapter 2, a detailed literature review of the work done in the area of Grid Security and addresses challenges including trust management and interoperability is presented.
- (ii) In chapter 3, three innovative interoperability solutions have been proposed to realise the original vision of grid to be used as a single massive infrastructure. The interoperability solutions proposed are Thapar University eXtensible - INTEROperability System (TUX-INTERO), Thapar University eXtensible Interoperability Portal (TUX-INTEROP) and Pluggable, Interoperable and Extensible (PIE). TUX-INTERO is a Web Services based interoperability solution which discovers and integrates resources based on various platforms and middleware. TUX-INTEROP is a grid portal for secure interoperation of grids. PIE is grid portal which integrates resources from various middleware and allows access to end users at a single interface.
- (iii) In chapter 4, a novel and flexible Reputation based Trust Management system which combines the merit of policy proof and reputation evaluation has been proposed. The trust management solution is Thapar University Extensible-Trust Management System (TUX-TMS) for establishing and evaluating trust among untrusted and unknown partners in a Grid. This system provides incentive to resource/service providers to contribute resources and motivates resource/service users to interact and transact depending on their reputation and trustworthiness information.

- (iv) The design, development and implementation of proposed Interoperability and Trust Management Solutions have been presented in Chapter 3 and 4, respectively.
- (v) In chapter 5, the thesis presents integration and testing results of TUX-INTERO and TUX-TMS on TUX-CITMS: A Grid Testbed which spans across Thapar University Campus using several test cases. It also demonstrates the testing results of other interoperability solutions, TUX-INTEROP and PIE using TU Campus Grid using a set of test cases. Every test case defines a testing criteria, test inputs, expected outputs and finally validates the outcome to be correct/ as expected.
- (vi) The testing results demonstrate the applicability of the proposed solutions in interoperating and securing transaction among multiple domains in a Grid environment.

6.2 Future Directions

Interoperability and Trust Management is a relatively young and complex research field with many different emerging ideas and solutions. This has led to a situation where there is confusion, ambiguity, conflicts, misinterpretations, an absence of core principles and a lack of standard approaches towards and about research in this field. In this thesis, a presentation is given on the basics of Interoperability and Trust Management in Grid Computing environments. It is hoped that the information presented will help to form a common core of knowledge. It can act as a starting point of research to design solutions for distributed environments such as Cloud Computing tomorrow. There is immense scope of research in this area. The work can be further enhanced

to enable Grid users and scientists to achieve and utilize true potential of this massive infrastructure- Grid. However, the work can be further enhanced and extended in future. Some of the future directions are:

- Cloud computing is the next generation paradigm for sharing computation or storage resources. It too faces Interoperability and Trust Management challenges that need to be addressed. TUX-INTERO, TUX-INTEROP, PIE and TUX-TMS can be extended to be included and used in Cloud Computing so as to enable usage and adoption of the same.
- The existing solutions have been designed and tested only for Interoperability and Trust Management challenges. In future, this can be extended for addressing the third security challenge, Integration, as identified by OGF.
- Currently, Interoperability and Trust Management solutions are middleware oriented which can be made to collaborate disparate Grids and secure the establishment of action trust and enable interaction between them.
 - TUX-INTERO can be extended to support other middleware such as gLite or NorduGrid/ARC which makes use of BDII for information services.
- Support for additional resource types: A large number of existing resource types, such as XGrid and other arbitrary nodes using the Secure Shell (SSH) can be supported. The support for other types of Grid resources such as heterogeneous data nodes can be achieved by extending the solution to use OGSA-DAI in future.
- Automating resource positioning: The resources can be geolocated based on their Internet Protocol (IP) or hostname or by using their supplied global positioning coordinates. It is possible for resources to transmit their global positioning coordi-

nates to TUX-INTERO while they are queried. Using this, the global positioning of resources could be more accurately plotted and monitored.

- TUX-TMS can be extended to incorporate more Policy based trust management system features to make establishment of trust more robust in a Grid.
- The behavior of the participants in Grid environments can be monitored and managed through the use of statistical methods.

References

- [1] G. Coulouris, J. Dollimore and T. Kindberg. Distributed Systems: Concepts and Design. Addison Wesley, *3rd edition*, (2000).
- [2] K. Nadiminti, M. Dias de Assuncao and R. Buyya. Distributed Systems and Recent Innovations: Challenges and Benefits. InfoNet Magazine, The University of Melbourne, Australia, Vol. 16(3), (2006), pp.1-5.
- [3] I. Foster and C. Kesselman. The Grid: A Blueprint for a new Computing Infrastructure. Morgan Kauffman Publications, *1st edition*, (1998).
- [4] I. Foster and C. Kesselman. The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufman Publishers, *2nd edition*, (2004).
- [5] M. Baker, R. Buyya and D. Laforenza. Grids and Grid technologies for wide-area distributed computing. Software Practice and Experience, Vol. 32(15), (2002), pp. 1437 - 1466.
- [6] M. M. Raghuwanshi and O. G. Kakde. Study of Operator's adaptability and scale-up study for RAGA. Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, (2007), pp. 2866-2871.
- [7] P. Mitra and C. Chaudhuri. Efficient Algorithm for the Extraction of Association Rules in Data Mining. Proceedings of Computational Science and Its Applications, LNCS, Springer-Verlag, Vol. 3981, (2006), pp. 1-10.
- [8] World Community Grid. Available Online at <http://www.worldcommunitygrid.org/>. Accessed on December 6, 2010.
- [9] A. Srinivasan, M. Mascagni and D.M. Ceperley. Testing parallel random number generators. Parallel Computing, Vol. 29, (2003), pp. 69-94.
- [10] A. Srinivasan and N. Chandra. Latency Tolerance through Parallelization of Time in Scientific Applications, Parallel Computing, Vol. 31, (2005), pp. 777-796.
- [11] J. Deogun, X. Lin, Y. Lu and S. Goddard. Real-time scheduling of divisible loads in cluster computing environments. Journal of Parallel and Distributed Computing, Vol. 70(3), (2010), pp. 296-308.

- [12] April J. Wells. Grid Database Design. Auerbach Publications, 1st edition, (2005).
- [13] F. Magoules, T. M. Huong Nguyen and L. Yu. Grid Resource Management. CRC Press, Chapman and Hall, 1st edition, (2009).
- [14] J. Joseph and C. Fellenstein. Grid Computing. Pearson Education, 1st edition, (2004).
- [15] David D. Roure, M.A. Baker, N.R. Jennings and N.R. Shadbolt. The evolution of the Grid. In: Grid Computing - Making the Global Infrastructure a Reality. John Wiley and Sons Ltd. Publishing, pp. 65-100, (2003).
- [16] FAFNER. Available Online at: <http://www.lehigh.edu/~bad0/fafner.html>. Accessed on April 10, 2010.
- [17] T. DeFanti, I. Foster, M. Papka, R. Stevens and T. Kuhfuss. Overview of the I-WAY: Wide Area Visual Supercomputing. International Journal of Supercomputer Applications, Vol. 10(2-3), (1996), pp. 123-131.
- [18] SETI@home: The search for extraterrestrial intelligence at home 2004. Available Online at: <http://setiathome.ssl.berkeley.edu/>. Accessed May 1, 2010.
- [19] Distributed.Net. Available Online at: <http://www.distributed.net>. Accessed May 1, 2010.
- [20] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. LNCS, Springer-Verlag, Vol. 3779, (2005), pp. 2-13.
- [21] A. Grimshaw, A. Ferrari, F. Knabe and M. Humphrey. Wide-area computing: Resource sharing on a large scale. IEEE Computing Practices, Vol. 32(5), (1999), pp. 29-37.
- [22] Condor. Available Online at <http://www.cs.wisc.edu/condor>. Accessed May 13, 2010.
- [23] Robert Henderson. Job scheduling under the portable batch system. LNCS, Springer-Verlag, Vol. 949, (1995), pp. 279-294.
- [24] Gridsphere. Available Online at <http://www.gridsphere.org/gridsphere/gridsphere>. Accessed on May 15, 2010.
- [25] J. Almond and D. Snelling. UNICORE: Uniform access to supercomputing as an element of electronic commerce. Future Generation Computer Systems, Vol. 15(5-6), (1999), pp. 539-548.
- [26] G. Allen, T. Dramlitsch, I. Foster, N. Karonis, M. Ripeanu, Ed Seidel and B. Toonen. Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. Proceedings of Supercomputing, Denver, USA, (2001), pp.1-25.

- [27] Open Grid Forum. Available Online at <http://www.ogf.org>. Accessed on May 13, 2010.
- [28] The Open Grid Services Architecture Version 1.0. Available Online at <http://www.gridforum.org/documents/GFD.30.pdf>. Accessed on February 10, 2010.
- [29] Web Services Resource Framework. Available Online at <http://www.ogf.org/documents/GFD.72.pdf>. Accessed on May 14, 2010.
- [30] Teragrid. Available Online at <https://www.teragrid.org>. Accessed on May 13, 2010.
- [31] Open Science Grid. Available Online at <http://www.opensciencegrid.org>. Accessed on May 15, 2010.
- [32] Enabling Grids for E-Science. Available Online at <http://public.eu-egee.org>. Accessed on May 15, 2010.
- [33] e-Science. Available Online at <http://www.nesc.ac.uk/>. Accessed on May 13, 2010.
- [34] K* Grid. Available Online at <http://www.gridcenter.or.kr>. Accessed on May 13, 2010.
- [35] NAREGI: National Research Grid Initiative. Available Online at http://www.naregi.org/index_e.html. Accessed May 13, 2010.
- [36] GARUDA. Available Online at <http://www.garudaindia.in>. Accessed May 13, 2010.
- [37] OurGrid. Available Online at <http://www.ourgrid.org>. Accessed on May 13, 2010.
- [38] Grid'5000. Available Online at <https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home>. Accessed on May 13, 2010.
- [39] AuverGrid. Available Online at <http://www.auvergrid.fr>. Accessed on May 13, 2010.
- [40] DAS: The Distributed ASCI Supercomputer 3. Available Online at <http://www.cs.vu.nl/das3>. Accessed on May 13, 2010.
- [41] B. Jacob, M. Brown, K. Fukui and N. Trivedi. Introduction to Grid Computing. IBM Redbooks. Available Online at: <http://www.redbooks.ibm.com/abstracts/sg246778.html>. Accessed on May 1, 2010.

- [42] L. Ferreira, V. Berstis, J. Armstrong and M. Kendzierski. Introduction to Grid Computing with Globus. IBM Redbooks. Available Online at: <http://www.redbooks.ibm.com/abstracts/sg246895.html>. Accessed on May 1, 2010.
- [43] PGRADE. Available Online at <http://portal.p-grade.hu/>. Accessed on May 15, 2010.
- [44] D. Abramson, P. Roe, L. Kotler and D. Mather. ActiveSheets: Super-Computing with Spreadsheets. Proceedings of High Performance Computing Symposium, Washington (USA), (2001), pp. 110 - 115 .
- [45] R. Barbera. The GENIUS Grid Portal. Proceedings of Computing in High Energy and Nuclear Physics, California, USA, (2003), pp. 1-8.
- [46] GridScape. Available Online at <http://www.cloudbus.org/gridscape/>. Accessed on February 19, 2011.
- [47] M. Placek and R. Buyya. G-Monitor: A Web Portal for Monitoring and Steering Application Execution on Global Grids. Proceedings of International Workshop on Challenges of Large Applications in Distributed Environments, Washington, USA, (2003), pp. 10-18.
- [48] R. Buyya, D. Abramson and J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. Proceedings of 4th International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, (2000), pp. 283-289.
- [49] Gridbus. Available Online at <http://www.gridbus.org/>. Accessed on May 20, 2010.
- [50] E. Elmroth and J. Tordsson. An interoperable standards-based Grid resource broker and job submission service. Proceedings of 1st IEEE Conference on e-Science Grid Computing, Melbourne, Australia, (2005), pp. 212-220.
- [51] I. Rodero, J. Corbalan, R. Badia and J. Labarta. eNANOS Grid Resource Broker. European Grid Conference, LNCS, Springer-Verlag, Vol. 3470, (2005), pp. 111-121.
- [52] A. Kertesz and P. Kacsuk. META-BROKER for Future Generation Grid: A New Approach for a High-Level Interoperable Resource Management. Proceedings of the Core GRID Workshop on Grid Middleware, Grid Middleware and Services, Springer, Vol I, (2008), pp. 53-63.
- [53] Condor-G. Available Online at <http://www.cs.wisc.edu/condor/condorg/>. Accessed on May 20, 2010.

- [54] A. Hurault and A. YarKhan. Intelligent Service Trading and Brokering for Distributed Network Services in GridSolve. Proceedings of 9th International Meeting on High Performance Computing for Computational Science, LNCS, Springer-Verlag, Vol. 6449, (2010), pp. 340-351.
- [55] E. Huedo, R. S. Montero and I. M. Llorente. A framework for adaptive execution in Grids. Software Practice and Experience, Vol. 34(7), (2004), pp. 631-651.
- [56] Sun Grid Engine. Available Online at <http://wikis.sun.com/display/GridEngine/Home>. Accessed on May 20, 2010.
- [57] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid. International Journal of High Performance Computing Applications, Vol.15 (3), (2001), pp. 200-222.
- [58] I. Foster, C. Kesselman, J. M. Nick and S. Tuecke. The Physiology of the Grid, An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure Working Group, Global Grid Forum (2002). Available Online at <http://www.globus.org/alliance/publications/papers/ogsa.pdf>. Accessed on July 10, 2010.
- [59] Hoon W. Lim. On the Application of Identity-Based Cryptography in Grid Security, PhD Thesis, University of London (2006).
- [60] M. Humphrey and Mary R. Thompson. Security Implications of Typical Grid Computing Usage Scenarios. Cluster Computing, Vol. 5(3), (2002), pp. 257-264.
- [61] H. Casanova. Distributed Computing Research Issues in Grid Computing. ACM SIGACT News Distributed Computing, Vol. 33(3), (2002), pp. 50-70
- [62] J. Nabrzyski, Jennifer M. Schopf and J. Weglars. Grid Resource Management: State of the Art and Future Trends. Kluwer Academic Publishers (2004).
- [63] L. Ramakrishnan. Securing Next Generation Grids. IT Professional. IEEE Computer Society, Vol. 6(2), (2004), pp. 34-39.
- [64] OGSA Security Roadmap. Available Online at <http://www.globus.org/toolkit/security/ogsa/>. Accessed on February 25, 2010.
- [65] P. J. Broadfoot and A. P. Martin. A Critical Survey of Grid Security Requirements and Technologies. Programming Research Group. Technical Report PRG-RR-03-15, Oxford University Computing Laboratory (2003).
- [66] M. Humphrey, Mary R. Thompson and Keith R. Jackson. Security for Grids. Proceedings of the IEEE, Vol. 93(3), (2005), pp. 644-652.
- [67] A. Chakrabarti. Grid Computing Security. Springer (2007).

- [68] A.Q.M. Khaliq, J. Martyn Vaquero, B.A. Wade and M. Yousuf. Smoothing Schemes for Reaction-Diffusion Systems with Nonsmooth Data. *Journal of Computer and Applied Mathematics*, Vol. 223(1), (2009), pp. 374-386.
- [69] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries (1990).
- [70] D Karpenko. Interoperability and Interoperations of Grid, *Proceedings of 21st CODATA Conference*, Kiev, Ukraine, (2008), pp. 241-243.
- [71] The Globus Toolkit. Available Online at <http://www.globus.org/toolkit>. Accessed on June 5, 2010.
- [72] The Transport Layer Security Protocol Version 1.0, IETF, 1999. Available Online at <http://www.ietf.org/rfc/rfc2246.txt>. Accessed on June 15, 2010.
- [73] Web Services Security. Available Online at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. Accessed on June 15, 2010.
- [74] Security Assertion Markup Language. Available Online at <http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTOKENProfile.pdf>. Accessed on June 15, 2010.
- [75] Extensible Access Control Markup Language. Available Online at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml. Accessed on June 15, 2010.
- [76] H.G. Hegering, S. Abeck and B. Neumair. *Integrated Management of Networked Systems: Concepts, Architectures and Their Operational Application*. Morgan Kaufmann, San Francisco (1999).
- [77] David W Chadwick, S. Otenko and V. Welch. Using SAML to link the GLOBUS toolkit to the PERMIS authorization infrastructure. *Proceedings of Conference on Communications and Multimedia Security*, Windermere, U.K., (2004), pp. 251-261.
- [78] D. W. Chadwick, A. Otenko and E. Ball. Implementing Role Based Access Controls Using X.509 Attribute certificates - the PERMIS Privilege Management Infrastructure. *IEEE Internet Computing*, Vol.7(2), (2003), pp. 62 - 69.
- [79] W. Xu, David W. Chadwick and S. Otenko. Development of a Flexible PERMIS Authorisation Module for Shibboleth and Apache Server. *Proceedings of EuroPKI*, England, (2005), pp. 162-179.
- [80] Shibboleth Architecture. Available Online at <http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-latest.pdf>. Accessed on February 19, 2011.

- [81] GridShib. Available Online at <http://gridshib.globus.org/about.html>. Accessed on June 21, 2010.
- [82] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker and Kate Keahey. Identity Federation and Attribute-Based Authorization through the Globus Toolkit, Shibboleth, GridShib and MyProxy. NIST PKI Workshop, April (2006). Available Online at <http://grid.ncsa.illinois.edu/papers/gridshib-pki06-final.pdf>. Accessed on February 19, 2010.
- [83] D.W. Chadwick, A. Novikov, and O. Otenko. GridShib and PERMIS Integration. *Campus-Wide Information Systems*, Vol. 23(4), (2006), pp. 297-308.
- [84] Shibboleth Enabled Bridge to Access the National Grid Service. Available Online at <http://www.rcs.manchester.ac.uk/research/shebangs>. Accessed on June 20, 2010.
- [85] Marcos D. de Assuncao, K. Nadiminti, S. Venugopal, T. Ma and R. Buyya. An Integration of Global and Enterprise Grid Computing: Gridbus Broker and Xgrid Perspective. *Proceedings of the 4th International Conference on Grid and Cooperative Computing, LNCS, Springer-Verlag*, Vol. 3795, (2005), pp. 406-417.
- [86] C. Pinchak, P. Lu, J. Schaeffer, M. Goldenberg. The Canadian Internetworked Scientific Supercomputer. *Proceedings of 17th International Symposium on High Performance Computing Systems and Applications, Quebec, Canada*, (2003), pp.193-199.
- [87] C. Pinchak, P. Lu, J. Schaeffer, M. Goldenberg. Practical Heterogeneous Placeholder Scheduling in Overlay Metacomputers: Early Experiences. *Proceedings of 8th Workshop on Job Scheduling Strategies for Parallel Processing, Edinburgh, Scotland, UK* (2002), pp. 85-105.
- [88] Workload Management with LoadLeveler. IBM Redbooks. Available Online at <http://www.redbooks.ibm.com/redbooks/pdfs/sg246038.pdf>. Accessed on June 20, 2010.
- [89] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. Van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schitt, E. Seidel, B. Ullmer. The grid application toolkit: toward generic and easy application programming interfaces for the grid. *Proceedings of the IEEE*, Vol.93(3), (2005), pp. 534-550.
- [90] G. Allen, T. Dramlitsch, I. Foster, Nicholas T. Karonis, M. Ripeanu, E. Seidel and B. Toonen. Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus. *Proceedings of Supercomputing, Denver, USA* (2001), pp. 52.
- [91] V. Mann and M. Parashar. Engineering an interoperable computational collaboratory on the Grid, *Concurrency and Computation: Practice and Experience* Vol. 14, (2002), pp. 1569-1593.

- [92] John M. Brooke, D. Fellows and J. MacLaren. Interoperability of Resource Description across grid domain boundaries. Proceedings of European Congress on Computational Methods in Applied Sciences and Engineering, Finland, (2004), pp 1-18.
- [93] F. Neubauer, A. Hoheisel, and J. Geiler. Workflow-based Grid applications. Future Generation Computer System, Vol. 22(1-2), (2006), pp. 6-15.
- [94] J. Cao, S. A. Jarvis, S. Saini and G. R. Nudd. GridFlow: WorkFlow management for Grid computing. Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, Tokyo, Japan, (2003), pp. 198-205.
- [95] G. Singh, G. Singh, E. Deelman, G. Mehta, K. Vahi, M. Su, G. B. Berrihan, J. Good, J. C. Jacob, and D. S. Katz. The Pegasus portal: Web based Grid computing. The 20th Annual ACM Symposium on Applied Computing, Santa Fe, New Mexico, March 13 -17, 2005. Available Online at http://pegasus.isi.edu/publications/Pegasus_Portal_final2.pdf. Accessed on February 20, 2010.
- [96] P. Kacsuk, T. Kiss, and G. Sipos. Solving the Grid interoperability problem by P-GRADE portal at workflow level. Future Generation Computer System, Vol. 24(7), (2008), pp. 744-751.
- [97] P. Kacsuk and G. Sipos. Multi-Grid, multi-user workflows in the P-GRADE Grid portal. Journal of Grid Computing, Vol. 3(3-4), (2006), pp. 221-238.
- [98] T. Kiss and T. Kukla. Achieving Interoperation of Grid Data Resources via Workflow Level Integration. Journal of Grid Computing, Vol. 7, (2009), pp. 355-374.
- [99] Open Grid Services Architecture Data Access and Integration. Available Online at <http://www.ogsadai.org.uk/>. Accessed on July 15, 2010.
- [100] Storage Resource Broker. Available Online at http://www.sdsc.edu/srb/index.php/Main_Page. Accessed on July 15, 2010.
- [101] Y. Zetuny, G. Kecskemeti, T. Kiss, G. Sipos, P. Kacsuk, G. Terstyanszky and S. Winter. Deployment and Interoperability of Legacy Code Services. Proceedings of the CoreGRID workshop, Pisa, Italy, (2005), pp 397-416.
- [102] Donno V. Ciaschini, D. Rebatto, L. Vaccarossa and M. Verlato. The World-Grid transatlantic testbed: a successful example of Grid interoperability across EU and U.S. domains. Proceedings of Computing in High Energy and Nuclear Physics, California, USA, (2003), pp. 24-28.
- [103] S. Venugopal, K. Nadiminti, H. Gibbins and R. Buyya. Designing a resource broker for heterogeneous Grids. Software Practice and Experience, Vol. 38(8), (2008), pp. 793-825.

- [104] A. Kertesz and P. Kacsuk. Grid Interoperability Solutions in Grid Resource Management. *IEEE Systems Journal*, Special Issue on Grid Resource Management, Vol. 3(1), (2009), pp. 131-141.
- [105] I. Rodero, F. Guim, J. Corbalan, L. L. Fong, Y. G. Liu and S. M. Sadjadi. Looking for an evolution of Grid scheduling: Meta-brokering, *Proceedings of Coregrid Workshop in Grid Middleware*, Dresden, Germany, (2007), pp.105-119.
- [106] N. Bobroff, L. Fong, S. Kalayci, Y. Liu, J.C. Martinez, I. Rodero, S.M. Sadjadi and D. Villegas. Enabling interoperability among metaschedulers. *Proceedings of Eighth IEEE International Symposium on Cluster Computing and the Grid*, Washington, USA, (2008), pp. 306-315.
- [107] G. Pierantoni, E. Kenny, B. Coghlan, O. Lyttleton D. O'Callaghan and G. Quigley. Interoperability using a Metagrid Architecture. *Proceedings of High Performance Distributed and Parallel Computing*, Paris, France, (2006), pp. 23-30.
- [108] M.M. Morgan and A.S. Grimshaw. Genesis II - standards based grid computing. *Proceedings of Seventh IEEE International Symposium on Cluster Computing and the Grid*, Rio De Janeiro, Brazil (2007), pp. 611-618.
- [109] HPC Basic Profile version 1.0, Available Online at <http://www.gridforum.org/documents/GFD.100.pdf>, Accessed on June 10, 2010.
- [110] Resource namespace service specification. Available Online at <http://www.ogf.org/documents/GFD.101.pdf>. Accessed on February 19, 2011.
- [111] ByteIO specification 1.0. <http://www.ogf.org/documents/GFD.87.pdf>. Accessed on February 19, 2011.
- [112] Implementation and interoperability experiences with the Job Submission Description Language (JSDL). Available Online at http://www.ogf.org/Public_Comment_Docs/Documents/2008-07/draft-gwde-jsdl-experience-1.0-007.pdf. Accessed on February 19, 2011.
- [113] Independent software vendors (ISV) remote computing usage primer. Available Online at <http://www.ogf.org/documents/GFD.141.pdf>. Accessed on February 20, 2010.
- [114] S. Andreozzi, S. Burke, L. Field, and B. Konya. Towards GLUE 2: evolution of the computing element information model. *Journal of Physics: Conference Series*, Vol. 119(6), (2008), p. 062009.
- [115] European Telecommunications Standards Institute. ETSI GRID. Available Online at: <http://www.etsi.org/WebSite/homepage.aspx>. Accessed on February 19, 2011.

- [116] OMII Europe. OMII Europe - open middleware infrastructure institute. Available Online at <http://www.omii-europe.org/>. Accessed on February 19, 2011.
- [117] EGEE's gLite - lightweight middleware for grid computing. Available Online at: <http://glite.cern.ch/>. Accessed on February 19, 2011.
- [118] UniGrids EU STREP project, European Commission. Available Online at <http://www.unigrids.org/>. Accessed on December 10, 2010.
- [119] C. Grimme, J. Lepping, A. Papaspyrou, P. W., R. Yahyapour, A. Oleksiak, O. Waldrich and W. Ziegler. Towards a standards-based Grid Scheduling Architecture. CoreGRID Technical Report Number TR-0123, Institute on Resource Management and Scheduling, December 31, 2007.
- [120] D. Kouril and J. Basney. A Credential Renewal Service for Long-Running Jobs. Proceedings of 6th IEEE/ACM International Workshop on Grid Computing, Washington, USA, (2005), pp. 63-68.
- [121] P. J. Strange, T. Antoni, F. Donno, H. Dres, G. Grein, G. Mathieu, A. Mills, D. Spence, T. Min and M. Verlati. Global GRID User Support : The Model and Experience in LHC Computing GRID. Available Online at <http://lcg.web.cern.ch/lcg/>. Accessed on September 5, 2010.
- [122] E. Elmroth and J. Tordsson. An Interoperable Standards-based Grid Resource Broker and Job Submission Service. Proceedings of First IEEE Conference on e-Science and Grid Computing, Melbourne, Australia, (2005), pp. 212-220.
- [123] O. Smirnova, P. Eerola, T. Ekelof, M. Ellert, J. R. Hansen, A. Konstantinov, B. Konya, J. L. Nielsen, F. Ould-Saada and A. Waananen. The NorduGrid architecture and middleware for scientific applications, LNCS, Springer-Verlag, Vol. 2657, (2003), pp.264-273.
- [124] T. Baur, R. Breu, T. Kalman, T. Lindinger, A. Milbert, G. Poghosyan, H. Reiser and M. Romberg. An Interoperable Grid Information System for Integrated Resource Monitoring Based on Virtual Organizations. Journal of Grid Computing Vol. 7, (2009), pp. 319-333.
- [125] M. Hassan and A. Abdullah. A New Grid Resource Discovery Framework. The International Arab Journal of Information Technology, Vol. 8(1), (2011), pp 20-29.
- [126] Open Grid Forum HPC Grid Interoperability Demonstration at SC06. Info about interoperability demos: <https://forge.gridforum.org/sf/wiki/do/viewPage/projects.ogsa-hpcp-wg/wiki/HomePage>. Accessed on February 25, 2010.

- [127] A. Iosup, D. H. J. Epema, T. Tannenbaum, M. Farrellee and M. Livny. Interoperating Grids through delegated matchmaking. Proceedings of ACM/IEEE Conference on Supercomputing, New York, USA, (2007), pp. 1-12.
- [128] OGSA Basic Execution Service version 1.0. Available Online at <http://www.ogf.org/documents/GFD.108.pdf>. Accessed on February 10, 2010.
- [129] R. Boardman, S. Crouch, H. Mills, S. Newhouse and J. Papay. Towards Grid interoperability. Proceedings of the UK e-Science All Hands Meeting 2007, Nottingham, UK, (2007), pp. 127-134.
- [130] Storage Resource Manager Interface Specification. Available Online at <http://www.ggf.org/documents/GFD.129.pdf>. Accessed on May 10, 2010.
- [131] R. Alfieri, R. Cecchini and V. Ciaschini. VOMS: an authorization system for virtual organizations. Proceedings of the First European Across Grids Conference, LNCS, Springer-Verlag, Vol. 2970, (2004), pp. 33-40.
- [132] S. Venugopal, R. Buyya and L. Winton. A Grid service broker for scheduling e-science applications on global data Grids. Concurrency Computation: Practice Experience, Vol. 18(6), (2006), pp. 685-699.
- [133] B. Boghosian, P. Coveney, S. Dong, L. Finn, S. Jha, G. Karniadakis and N. Karonis. Nektar, SPICE and Vortronics - Using Federated Grids for Large Scale Scientific Applications. Cluster Computing, Vol. 10(3), (2007), pp. 351-364.
- [134] B. Bethwaite, D. Abramson and A. Buckle. Grid Interoperability: An Experiment in Bridging Grid Islands. Proceedings of Fourth IEEE International Conference on eScience, Indiana, USA, (2008), pp. 590-596.
- [135] Y. Cheng, G. Chen, Y. Wang and S. Wang. Deploying HEP Applications on Multiple Grid Infrastructures. Proceedings of Seventh International Conference on Grid and Cooperative Computing, Shenzhen, China, (2008), pp. 632-641.
- [136] L. Field and E. Laure. Towards Seamless Grid Computing. Mem. S. A. It. Vol. 80, (2009), pp. 497-503.
- [137] L. Field, S. Andreozzi and B. Konya. Grid Information System Interoperability: The Need For A Common Information Model. Proceedings of Fourth IEEE International Conference on eScience, Indianapolis, USA, (2008), pp. 501-507.
- [138] ETSI. GRID: Study of ICT Grid interoperability gaps. Technical report, European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France (2008).
- [139] GRID. ICT Grid Interoperability Testing Framework and survey of existing ICT Grid interoperability solutions. European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France (2009). Available Online

- at http://www.etsi.org/deliver/etsi_tr/102700_102799/102766/01.01.01_60/tr_102766v010101p.pdf. Accessed on February 10, 2011.
- [140] C. Davenhall and M. Mineter. Standards for Interoperable Grids. Experience from NextGRID and OMII-Europe, 17 March 2008, version 1 (2008). Available Online at <http://www.nesc.ac.uk/talks/869/StandardsforInteroperableGrids.pdf>. Accessed on February 10, 2011.
- [141] Ninf. Available Online at <http://ninf.apgrid.org/>. Accessed on June 10, 2010.
- [142] M. Flechl and L. Field. Grid interoperability: Joining Grid information systems. *Journal of Physics: Conference Series*, Vol. 119, (2008), pp. 1-7.
- [143] L. Field, E. Laure and Markus W. Schulz. Grid Deployment Experiences: Grid Interoperation. *Journal of Grid Computing*, Vol. 7, (2009), pp. 287-296.
- [144] Grid Interoperation Now Community Group (GIN-CG). Available Online at <http://forge.ogf.org/sf/projects/gin>. Accessed on May 26, 2010.
- [145] J. Jensen, R. Downing, D. Ross, M. Hodges and Alex Sim. Practical Grid Storage Interoperation: Interoperation of SRM and SRB Now. *Journal of Grid Computing*, Vol. 7(3), (2009), pp. 309-317.
- [146] An Introduction to Cryptography, in PGP 6.5.1 User's Guide, Network Associates Inc. p. 11 - 36. Available Online at <http://www.pgpi.org/doc/pgpintro/>. Accessed on June 11, 2010.
- [147] C. Adams and S. Farrell. RFC2510 - Internet X.509 Public Key Infrastructure Certificate Management Protocols. Available Online at <http://www.ietf.org/rfc/rfc2510.txt>. Accessed on June 10, 2010.
- [148] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Available Online at <http://www.ietf.org/rfc/rfc3280.txt>. Accessed on June 12, 2010.
- [149] M. Blaze, J. Feigenbaum and J. Lacy. Decentralized Trust Management. *Proceedings of the 17th IEEE Symp. on Security and Privacy*. IEEE Computer Society (1996), pp 164-173. Available at <http://www.crypto.com/papers/policymaker.pdf>. Accessed on June 16, 2010.
- [150] C. Dellarocas, F. Dini and G. Spagnolo. Designing Reputation Mechanisms. *Handbook of Procurement*, Cambridge University Press, (2007), pp. 446-482.
- [151] J. Miller, P. Resnick and D. Singer. PICS Rating Services and Rating Systems (and Their Machine Readable Descriptions) version 1.1. Available Online at: <http://www.w3.org/TR/REC-PICSRules/>. Accessed on February 16, 2011.

- [152] M. Blaze, J. Feigenbaum, J. Ioannidis and A. Keromytis. Using the KeyNote Trust Management System, Request for Comments: 2704.
- [153] Y Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, M. Strauss. REF-EREE: Trust management for web applications. *World Wide Web Journal*, Vol. 2 (1997), pp. 127-139.
- [154] T. Jim. SD3: a trust management system with certified evaluation. *Proceedings to IEEE Symposium on Security and Privacy*, Oakland, California, USA (2001), pp. 106-115.
- [155] W. T. Ming Yao. Fidelis: A Policy-Driven Trust Management Framework. *LNCS*, Springer-Verlag, Vol. 2692, (2003), pp. 301-317.
- [156] W. Winsborough, K. Seamons and V. Jones. Automated Trust Negotiation: Managing Disclosure of Sensitive Credentials. *Transarc Research White Paper*, May 1999.
- [157] T. Barlow, A. Hess and K.E. Seamons. Trust Negotiation in Electronic Markets. *Proceedings of Eighth Research Symposium in Emerging Electronic Markets*, Maastricht, Netherlands (2001), pp.1-13.
- [158] Trusted Computing Platform Alliance. TCPA Design Philosophies and Concepts, Version 1.0. Available Online at <http://www.trustedcomputinggroup.org>. Accessed on June 18, 2010.
- [159] R. Chen, W. Yeager. Poblano: A distributed trust model for peer-to-peer networks. Technical report, Sun Microsystems. (2003). Available Online at http://www.cc.gatech.edu/classes/AY2002/cs8803d_spring/papers/P2P.pdf. Accessed on June 25, 2010.
- [160] N. Li, John C. Mitchell and William H. Winsborough. Design of a Role-Based Trust Management Framework. *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, California, USA, (2002), pp. 114-130.
- [161] H. Shrobe, J. Doyle and P. Szolovits. Active Trust Management for Autonomous Adaptive Survivable Systems. *DARPA Proposal*, MIT, AI Lab and Lab for Computer Science (2000), pp. 27.
- [162] L. Kagal, S. Cost, T. Finin and Y. Peng. A Framework for Distributed Trust Management. *Proceedings of Second Workshop on Norms and Institutions in MAS, Autonomous Agents*, Montreal, Canada (2001). Available Online at <http://ccs.mit.edu/dell/aa2001/aa2001papers/paper4.pdf>. Accessed on July 10, 2010.
- [163] M. A. Maarof and K. Krishna. An Hybrid Trust Management Model for MAS Based Trading Society. *The International Arab Journal of Information Technology*, Vol. 1, (2003), pp. 60-68.

- [164] Phill P. Zimmermann. The Official PGP User's Guide. MIT Press, Cambridge, MA (1995).
- [165] K. Aberer. P-Grid: A Self-Organizing Access Structure for P2P Information Systems. In Proceedings of Ninth International Conference on Cooperative Information Systems, LNCS, Springer-Verlag, Vol. 2172, (2001), pp. 179-194.
- [166] Sepandar D. Kamvar, Mario T. Schlosser and Hector G. Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. Proceedings of Twelfth International World Wide Web Conference, Budapest, Hungary, (2003), pp. 640-651.
- [167] Z. Abrams, R. McGrew and S. Plotkin. A non-manipulable trust system based on EigenTrust. ACM SIGecom Exchanges, Vol. 5(4), (2005), pp.21-30.
- [168] B. Dragovic and E. Kotsovinos. XenoTrust: Event-based distributed trust management. Proceedings of Second International Workshop on Trust and Privacy in Digital Business, Prague, Czech Republic, (2003), pp. 410-441.
- [169] B. Dragovic, S. Hand, T. Harris and E. Kotsovinos. Managing trust and reputation in the XenoServer Open Platform. Proceedings of the 1st International Conference on Trust Management, LNCS, Springer-Verlag, Vol. 2692, (2003), pp. 59-74.
- [170] L. Xiong and L. Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. IEEE Transactions on Knowledge and Data Engineering, Vol. 16(7), (2004), pp. 843-857.
- [171] S. Song, K. Hwang and M. Macwan. Fuzzy Trust Integration for Security Enforcement in Grid Computing. Proceedings of IFIP International Conference on Network and Parallel Computing, LNCS, Springer-Verlag, Vol. 3222, (2004), pp. 9-21.
- [172] S. Song, K. Hwang and Y.K Kwok. Trusted Grid Computing with Security Binding and Trust Integration. Journal of Grid Computing, Vol. 3(1), (2005), pp. 24-34.
- [173] B.E. Alunkal. GridEigenTrust: A Framework for Computing Reputation in Grids. MSc Thesis. Illinois Institute of Technology, Chicago, Illinois, USA (2003).
- [174] F. Kerschbaum, J. Haller, Y. Karabulut and P. Robinson. Pathtrust: A trust-based reputation service for virtual organization formation. Proceedings of the 4th International Conference on Trust Management, LNCS, Springer-Verlag, Vol. 3986, (2006), pp. 193-205.
- [175] S. Lee, R. Sherwood and B. Bhattacharjee. Cooperative peer groups in NICE. Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 50(4), (2006), pp. 523 - 544.

- [176] A. Abdul Rahman and S. Hailes. Supporting trust in virtual communities. Proceedings of Hawaii International Conference on System Sciences, Hawaii (2000), pp. 6007.
- [177] A. Singh and Ling Liu. Trustme: Anonymous management of trust relationships in decentralized p2p systems. Proceedings of 3rd International Conference on Peer-to-Peer Computing, Washington, USA (2003), pp. 142-149.
- [178] R. Levien. Attack Resistant Trust Metrics. Computing with Social Trust, Human Computer Interaction Series, 1st Edition, (2009).
- [179] C. N. Ziegler and G. Lausen. Spreading activation models for trust propagation. Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service, New York, USA, (2004), pp. 83-97.
- [180] L. Page, S. Brin, R. Motwani and T. Winograd. The PageRank citation ranking: Bringing order to the web. Proceedings of 7th International World Wide Web Conference, Brisbane, Australia, (1998), pp. 161-172.
- [181] A. Josang, R. Ismail and C. Boyd. A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems, Vol. 43(2), (2007), pp. 618-644.
- [182] G. Shafer. Perspectives on the theory and practice of belief functions. International Journal of Approximate Reasoning, Vol. 6(3), (1992), pp. 445-480.
- [183] J. Sabater and C. Sierra. Review on computational trust and reputation models. Kluwer Academic Publishers, Vol. 10(34), (2005), pp. 1-28.
- [184] F. Azzedin and M. Maheswaran. Evolving and Managing Trust in Grid Computing Systems. Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, Canada, (2002), pp. 1424-1429.
- [185] F. Azzedin and M. Maheswaran. Trust Modeling for Peer-to- Peer based Computing Systems. Proceedings of the International Parallel and Distributed Processing Symposium, Nice, France (2003), pp. 99a.
- [186] S. D. Ramchurn, T. D. Huynh and N. R. Jennings. Trust in Multiagent Systems. The Knowledge Engineering Review, Vol. 19(1), (2004), pp. 1-25.
- [187] F. Almenarez, A. Maryn, C. Campo, C. Garcya. PTM: a Pervasive Trust Management model for dynamic open environments. Privacy and Trust. First Workshop on Pervasive Security and Trust, Boston, MA, USA, (2004).
- [188] Z. Liang and W. Shi. PET: A Personalised Trust Model with Reputation and Risk Evaluation for P2P Resource sharing. Proceedings of the 38th Hawaii International Conference on System Sciences, Hawaii, (2005), pp. 201b.

- [189] G. C. Silaghi, A. E. Arenas, L. M. Silva. Reputation-based trust management systems and their applicability to Grids. Technical Report Number TR-0064. CoreGRID (2007).
- [190] P. Resnick, R. Zeckhauser, J. Swanson and K. Lockwood. The value of reputation on ebay: A controlled experiment. *Experimental Economics*, Vol. 9(2), (2006), pp.79-101.
- [191] R. Dingledine, M.J. Freedman and D. Molnar. Accountability Measures for Peer-to-Peer Systems. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly Publishers (2000).
- [192] F. L. Marmol and G. M. Perez. Security threats scenarios in trust and reputation models for distributed systems. *Computers and Security*, Vol. 28(7), (2009), pp. 545-556.
- [193] David O Callaghan. Trust Evaluation for the Grid. PhD Thesis, University of Dublin, Trinity College, (2007).
- [194] C. Lin, V. Varadharajan, Y. Wang and V. Pruthi. Enhancing Grid Security with Trust Management. *Proceedings of the IEEE International Conference on Services Computing*, Shanghai, China, (2004), pp. 303-310.
- [195] Microsoft Expands Passport to Enable Universal Single Sign-In. Available Online at <http://www.microsoft.com/presspass/press/2001/Sep01/09-20PassportFederationPR.mspx>. Accessed on June 2, 2010.
- [196] Entrust TruePass Product Portfolio: Technical Overview. White Paper. Available Online at <http://download.entrust.com/resources/download.cfm/21136>. Accessed on June 2, 2010.
- [197] Jason Novotny, Steven Tuecke and Von Welch. An Online Credential Repository for the Grid: MyProxy. *Proceedings of IEEE International Symposium on High Performance Distributed Computing*, San Francisco, USA (2001), pp. 104-111.
- [198] KX.509/KCA. Available Online at <http://www.citi.umich.edu/techreports/reports/citi-tr-01-2.pdf>. Accessed on June 2, 2010.
- [199] O. Kornievskaja, P. Honeyman, B. Doster and K. Coffman. Kerberized Credential Translation: A Solution to Web Access Control. *Proceedings of the 10th conference on USENIX Security Symposium*, Washington, USA, (2001), pp. 235-250.
- [200] B. Schneier. Password Safe: The Security of Blowfish in a Password Database. <http://www.schneier.com/passsafe.html>. Accessed on June 2, 2010.
- [201] Microsoft Windows 'TrustBridge' to enable organizations to share user identities across business boundaries (2002) Available Online at <http://www.microsoft.com/presspass/press/2002/Jun02/06-06TrustbridgePR.mspx>. Accessed on June 2, 2010.

- [202] J. Kemp, Ed. Liberty ID-WSF. May (2004). Available Online at [http://projectliberty.org/liberty/content/download/1236/8012/file/liberty-idwsf-1.0-scr-v1.0-08% 20Final.pdf](http://projectliberty.org/liberty/content/download/1236/8012/file/liberty-idwsf-1.0-scr-v1.0-08%20Final.pdf). Accessed on June 5, 2010.
- [203] David D. Vecchio, J. Basney and N. Nagaratnam. CredEx: User Centric Credential Management for Grid and Web Services. Proceedings of IEEE International Conference on Web Services, Florida, USA (2005), pp. 149 - 156.
- [204] Mark O Neill, Phillip H. Baker, Sean M. Cann, M. Shema, Ed Simon, Paul A. Watters and Andrew White. Web Services Security, McGraw-Hill Professional, (2003).
- [205] R. Sandhu, M. Bellare and R. Ganesan. Password-Enabled PKI: Virtual Smart Cards versus Virtual Soft Tokens. Proceedings of First Annual PKI Research Workshop Gaithersburg, Maryland, USA, (2002), pp. 89-96.
- [206] V. Welch, K. Keahey, F. Siebenlist and T. Barton. NMI development: Policy Controlled Attribute Framework - A Privacy Friendly Framework for Policy Enforced Release, Sharing, and Matching of Attribute Information, Project Summary, National Science Foundation, National Middleware Infrastructure Solicitation NSF 04-555, (2004). Available Online at <http://grid.ncsa.illinois.edu/GridShib/GlobusShibNMI04-Public.pdf>. Accessed on June 2, 2010.
- [207] The Globus Security. Available Online at <http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf>. Accessed on June 5, 2010.
- [208] D. Jana and B B Bhaumik. Single Signon for Grid Services. Proceedings of the IEEE INDICON, Kharagpur, India, (2004), pp. 513 - 516.
- [209] M. Lorch, J. Basney and D. Kafura. A Hardware-secured Credentials Repository for Grid PKIs. Proceedings of IEEE International Symposium on Cluster Computing and the Grid, Chicago, Illinois, USA (2004), pp. 640-647.
- [210] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer and C. Kesselman. A National-Scale Authentication Infrastructure. IEEE Computer, Vol. 33(12), (2000), pp. 60-66.
- [211] H. R. Nagesh, K. Chandra Sekaran and K. M. Hebbar. Design, implementation and performance analysis of secure group communication. Proceedings of the 15th International Conference on Computer Communication, Washington, USA, (2002), pp. 880 - 891.
- [212] M. Thompson, A. Essiari and S. Mudumbai. Certificate-based Authorization Policy in a PKI Environment. ACM Transactions on Information and System Security, Vol. 6(4), (2003), pp. 566 - 588.

- [213] L. Pearlman, V. Welch, I. Foster, C. Kesselman and S. Tuecke. A Community Authorization Service for Group Collaboration. Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, Monterey, California (2002), pp. 0050.
- [214] R. Lepro. Cardea: Dynamic Access Control in Distributed Systems. NAS Technical Report NAS-03-020 (2003). Available online at <http://www.nas.nasa.gov/News/Techreports/2003/PDF/nas-03-020.pdf>. Accessed on June 10, 2010.
- [215] M. Lorch, D.B. Adams, D. Kafura, M.S.R. Koneni, A. Rathi and S. Shah. The PRIMA System for Privilege Management. Authorization and Enforcement in Grid Environments. Proceedings of the Fourth International Workshop on Grid Computing, Phoenix, Arizona, USA, (2003), pp. 109.
- [216] R. Baker, D. Yu and T. Wlodek. A Model for Grid User Management. Computing in High Energy and Nuclear Physics, La Jolla, California, USA, (2003), Available Online at <http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUBT002.PDF>. Accessed on July 10, 2010.
- [217] G. Zhang and M. Parashar. SESAME: Scalable, Environment Sensitive Access Management Engine. Cluster Computing, Vol. 9(1), (2006), pp. 19-27.
- [218] S. Cannon, S. Chan, D. Olson, C.Tull, V. Welch and L. Pearlman. Using CAS to Manage Role-Based VO Sub-Groups. Proceedings of Computing in High Energy and Nuclear Physics , La Jolla, California, USA, (2003), Available Onlien at <http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUBT004.PDF>. Accessed on August 12, 2010.
- [219] H. Jin, W. Qiang, X. Shi and D. Zou. RB-GACA: A RBAC based grid access control architecture. International Journal of Grid and Utility Computing, Vol. 1(1), (2005), pp. 61-70.
- [220] NASA. Available Online at <http://ti.arc.nasa.gov/opensource/projects/ipg/>. Accessed on June 25, 2010.
- [221] Usage of LDAP in Globus. Available Online at http://www.cs.ucy.ac.cy/cross-grid/cygriddl/ldap_in_globus.pdf. Accessed on September 12, 2010.
- [222] UNICORE. Available Online at <http://www.unicore.eu/unicore/architecture.php>. Accessed on June 1, 2010.
- [223] LEGION. Available Online at <http://legion.virginia.edu/>. Accessed on June 1, 2010.
- [224] F. Gagliardi, B. Jones, M. Reale and S. Burke. European DataGrid project: Experiences of deploying a large scale testbed for e-science applications. Proceedings of Performance Evaluation of Complex Systems: Techniques and Tools. LNCS Springer-Verlag, Vol. 2459, (2002), pp. 480-501.

- [225] J. Gomes, M. David, J. Martins, L. Bernardo, A. Garcya, M. Hardt et. al. Experience with the international testbed in the CrossGrid project. Proceedings of Advances in Grid Computing EGC 2005, LNCS Springer-Verlag, Vol. 3470, (2005), pp. 98-110.
- [226] Large Hadron Collider Computing Grid Project. Available Online at <http://lcg.web.cern.ch/LCG/>. Accessed on September 12, 2010.
- [227] K. Stoupa and A. Vakali. Policies for Web Security Services. Available Online at <http://citeseer.ist.psu.edu/stoupa05policies.html>. Accessed on June 20, 2010.
- [228] S. Singh and S. Bawa. A Privacy, Trust and Policy based Authorization Framework for Services in Distributed Environments. International Journal of Computer Science (IJCS), Vol. 2(2), (2007), pp. 85-92.
- [229] GeoMaint. Available Online at <http://geomaint.sourceforge.net/>. Accessed on June 15, 2010.
- [230] Open Source Host-based Intrusion Detection System. Available Online at <http://www.ossec.net/>. Accessed on May 10, 2010.
- [231] Chongjie Zhang, Ian Kelley and Gabrielle Allen. Grid portal solutions: A comparison of GridPortlets and OGCE, Concurrency Computation: Practice and Experience, Volume 19, (2007), pp. 1739-1748.

List of Publications

Referred Journals

1. Shashi, Seema Bawa. Pluggable, interoperable and extensible (PIE): A grid portal for interoperability. International Journal of Physical Sciences. Volume 22, Issue 6 (2011). Pages 5329 - 5344(SCI Indexed, IF:
2. Shashi, Seema Bawa. Enhancing Reputation in a trust based system, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Hiedelberg, (2010), Pages 440-451.
3. Shashi, Seema Bawa. TUX-TMS (Thapar University Extensible-Trust Management System), International Journal of Security, CSC Journals, Volume 4, Issue 1, (2010), Pages 1-16.
4. Shashi, Seema Bawa. TUX-INTERO: A Portal for secure interoperation of Grids, International Journal of Engineering Science and Technology, Volume 2 Issue 7, (2010), Pages 3335-3343.
5. Shashi, Seema Bawa (2010). Interoperability: Bridging the Gap, International Journal of Computer Applications, Volume 7 No. 12, Pages 15-21.

International Conferences

6. Shashi, Seema Bawa, Establishing and Evaluating Trust in a Grid Environment, GridCAT Workshop, in the Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, Taiwan, December 9-11, (2009), Pages 674-678.
7. Shashi, Seema Bawa, Evaluating Trust in a Grid Environment, Student Research Symposium at HiPc (International Conference on High Performance Computing) (2008), Bangalore, India on December 17-20, 2008. Available at <http://www.hipc.org/hipc2008/documents/HiPC-SS08-FinalPapers/1569154841.pdf>.
8. Shashi, Seema Bawa, Managing Trust and Interoperability in Grid, New Investigator Technical Category, International Conference GHC-2008 (Grace Hooper celebration of women in computing), in Colorado, USA on October 1-4, (2008). Available at <http://gracehopper.org/2008/assets/GHC2008-Program.pdf>.
9. Shashi, Seema Bawa, Securing A Grid, International Conference CGCS-2008 (Cluster and Grid Computing Systems) By Waset (World academy of science, engineering and technology), Singapore, Aug 30-September 1, 2008, ISSN: 2070-3740, (2008), Pages 7-12.

Papers Communicated

1. Shashi, Seema Bawa. A novel hybrid recommendation system for open distributed environments. RecSys 2012.
2. Shashi, Seema Bawa. Multi constraints QoS parameters for establishing secure Grids. IET Information Security.
3. Shashi, Seema Bawa. TUX-CITMS: Interoperability and Trust Management Solution for Establishing Secure Grids, Journal of Network and Computing Applications.

Recognition for Work done

1. **BROADER ENGAGEMENT ADVANCE TRACK 2011 for SC-2011**
Awarded Broader Engagement Advance Track for SuperComputing (SC-2011) which included travel and accommodation for attending SC-2011, an International Conference on High Performance Computing, Networking, Storage and Analysis.
2. **GOOGLE INDIA WOMEN IN ENGINEERING AWARD 2010**
Awarded Special Commendation Award for making it to two editions of Google India Women in Engineering Award and in recognition of academic excellence and contribution to Computing and Technology.
3. **RISE 2009 (Research Internships in Science and Engineering), An Indo-US Science Technology Forum Initiative.**
Chosen Finalist for Research Internships in Science and Engineering from India.
4. **GOOGLE INDIA WOMEN IN ENGINEERING AWARD 2009**
Nominated for Google India Women in Engineering Award 2009 in recognition of academic excellence and contribution to Computing and Technology. Recognized as top Computer Engineering candidate in India by Google.
5. **HiPC SCHOLARSHIP 2008**
Awarded HiPc Travel Scholarship which included travel and accommodation for presenting my research paper titled “Evaluating Trust in Grid Environment” under Student Research Symposium where the selection rate was 40% for selecting papers for the Symposium.
6. **GOOGLE COMMUNITY SCHOLARSHIP 2008**
Awarded Google Community Scholarship for presenting Research paper entitled “Managing Trust and Interoperability in Grid” and attending ANITA BORG Women in Technology, GHC-2008 International Conference which includes travel, registration and accommodation. The Research paper was selected in “New Investigator Technical Category”.
7. **CDAC INDIAN STUDENT SCHOLARSHIP 2007**
Awarded CDAC Indian Student Scholarship for attending eScience 2007 (An International Conference on Grid Computing).