# Efficient Resource Discovery in Grid Environments

*A Thesis submitted*

*For the award of the degree of*

*DOCTOR OF PHILOSOPHY*

*by*

**Anju Sharma**

**(9050301)**

*under the Guidance of*

**Dr. Seema Bawa**

**Professor & Head**

**Department of Computer Science and Engineering**

**Thapar University, Patiala -147004**


**Computer Science and Engineering Department**

**Thapar University, Patiala – 147 004, INDIA**

**June, 2009**

*dedicated to*

*My Parents*

*whose sacrifices, love & affection*

*made me to capable of carrying out this work*

# Contents

# Certificate

I hereby certify that the work which is being presented in this thesis entitled *Efficient Resource Discovery in Grid Environments*, for the award of degree of *Doctor of Philosophy* submitted to the **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 researchers 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.

(Anju Sharma)
Reg. No. 9050301

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 & Head
Department of Computer Science & Engineering
Thapar University, Patiala 147 004, Punjab, INDIA

# Acknowledgements

Research endeavor and attainment of parsimony in research area, undoubtedly, is a team work. This is an opportunity to thank all the people who made this thesis possible and have helped me in completing this piece of research.

It gives me immense pleasure to express my sincerest gratitude to my supervisor, Dr. Seema Bawa, Professor and Head, Computer Science and Engineering Department, Thapar university, for her expert guidance and constant encouragement. I also appreciate the freedom she gave me with my working hours. I am deeply indebted to her for detailed, meticulous and repeated corrections of the manuscript of this thesis. Her ardent dedication to perfection and exactness inspired me to complete this thesis as good as possible.

I am immensely grateful to Dr. Maninder Singh, Assistant Professor, Computer Science and Engineering Department, Thapar University, a nice person and an excellent mentor, who always encouraged me to keep going with work and always advised me with his invaluable suggestions.

I would like to express my sincere thanks to Doctoral Committee Members, Dean of Student Affairs, Dr. R.K. Sharma, Assistant Professor Dr. Rajesh Kumar,

and Dean of Research and Sponsored Projects, Dr. Sushil Mittal, for their critical observations and valuable comments which helped enormously in presenting results and shaping this thesis.

I would like to thank faculty and staff members of Computer Science and Engineering Department of Thapar University, Patiala for providing and sharing resources that have been utilized in implementation of the Algorithms. I would also like to thank the Thapar University, Patiala for providing me financial assistance by awarding me Teaching Assistant fellowship during the period of my Ph.D.

This research has profited from the friendship, advice, encouragement and support of several remarkable people. Inderpreet Chopra without any doubt, top of this list for his continual counsel, willingness to listen, considered insights and enthusiasm. Ms. Inderveer Chana, Mr. Sarabjeet, Mr. Bhupinder, Ms. Anuradha, Mr. Pankaj Sharma, and Mr. Abhishek Sharma whose experienced guidance helped me to overcome many practical as well as technical issues.

Most importantly, I wish to thank my family. Pitaji, one of supporter for me throughout my studies and moral support. My deep regard to Mummy (Ms. Meena Sharma), Papa (Mr. K.N. Sharma), brothers: Er. Virender, Ravinder, Sonu & Pranav; Sisters in Laws: Ms. Kanchan, Dr. Ritu & Monika Rishi; Sisters Manju, Savita, Kavita, Sukriti, Priynaka & Rushal and Brother in Law: Sunil Joshi & Satvinder Joshi for their affection and good wishes. I am grateful to my lovely Nephews: Tushar, Saksham and Nieces: Niharika, Upasna, Sanvi, Vanshika and Isha for their love and affection. I am also thankful to my friend Neha, who with a gesture of family touch always encouraged me to complete my Ph.D.

I give my sincere thanks to my in-laws whose support made this thesis possible. Since words are the way to express the feelings so, I owe a special thanks to my husband, Ravinder Singh Joshi, for his love and unconditional support of all my undertakings, scholastic and support otherwise.

Last but not the least to God almighty that guides our lives and blessed me with strength to overcome every obstacle in this journey.

Anju Sharma
June 2009

# List of Figures

# List of Tables

# Abstract

Grid Computing has evolved into an important discipline by differentiating itself from distributed computing through an increased focus on resource sharing, coordination, manageability and high performance. Grid computing combines open, shared, geographically distributed and heterogeneous resources to achieve high computational performance. The objective of the Grid Computing is to solve large problems which can not be solved by single CPU by achieving high computing performance by optimal use of geographically distributed heterogeneous idle resources. These resources may belong to different institutions, different domains; may have different usage policies and may pose different requirements on acceptable requests. One of the major challenges in such highly heterogeneous and complex computing environments is to design efficient Resource Discovery algorithms. Resource Discovery strategies in such cases should be efficient, robust, and scalable.

Resource heterogeneity domains, dynamic load on resources, task runtime prediction uncertainty, task-to-resource ratio and resource sharing in the grid environment affects application performance. Grid resources are heterogeneous due to differences in hardware components, differences in Grid software environments, and due to the fact different administrative have different policies for sharing the resources.

This work mainly focuses on "Efficient Resource Discovery with heterogeneous resources". Initially, an in-depth review of existing models, approaches and algorithms has been done. During the literature review, it is observed that query based, agent based, parameter based and ontology based approach are some of the existing approaches. Flooding algorithm, swamping algorithm, name-dropper algorithm and kutten peleg algorithms. Three models; push, pull and push-pull models are currently being used for Resource Discovery process. A comprehensive study of different middleware also has been done. Different QoS parameters have been identified and analyzed as: Time, Cost, Reliability and Security. Time is the measure of delivery period of resources. Cost is the amount payable for using the resource or a set of resources including communication cost and computation cost. Penalty cost is included in case of delay in delivery of the resources required. Communication and computation costs are other costs. Reliability is about making delivery sure for a resource or a set of resources. Security is a mechanism for ensuring the trust while sharing or exchanging a resource among users or users and manager.

Three algorithms have been proposed for Resource Discovery namely: "Path Optimization Algorithm, "Weight Optimization Algorithm and "Request Matching Algorithm". Path Optimization Algorithm is based on the concept of "connected graphs" and their trees. This algorithm optimizes the path traversed for the Resource Discovery process. Weight Optimization Algorithm is based on the concept of directed weighted graph and it optimizes the total cost/total time needed for Resource Discovery process. A method of parallelizing both these algorithms is also described in detail. Request Matching Algorithm is designed based on the ontology approach and it finds the best resource from a set of all matching resources for a given job. It works with the three matchmaking principles. First - to find the acceptable matches; second - to supervise and modernize lists of the requester and provider and third - dynamically create the data updating in the resource pool, for

finding the acceptable matches. Complexity analysis as well as comparative analysis of all the three algorithms has been made.

To test and validate the proposed algorithms, a customized testbed has been developed as part of this work, namely TUGrid testbed. It has been used to demonstrates the usefulness of this work. Different test cases including different parameters for the Resource Discovery process have been considered for the evaluation of the proposed algorithms. The results clearly demonstrate that the approach is workable and can be effectively used to address the different QoS requirements including Time, Cost, Reliability and Security.

# 1

# Introduction

In today's complex world of high speed computing, computers are becoming extremely powerful and even home-based workstations are powerful enough for running complex applications [Dur04]. At the same time there is an increased demand for computational power for several complex scientific experiments in the field of advanced modeling, genome matching, astronomical research, real-time personnel portfolio management *etc.* that, widen the scope of computing. In this context, it requires huge amount of computational resources. Therefore, to satisfy the some of these aforementioned computational requirements, Parallel Computing came into existence.

Parallel Computing is the simultaneous execution of the same task on a computer with multiple processors which is capable of parallel processing. Parallel processing is used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of computer systems [Man93]. Various examples of parallel computing includes: weather and ocean patterns, daily operations within a business, management of national and multi-national corporations, array processing *etc.* It has given birth to distributed systems where complex problems are divided and allotted to parallel or distributed systems so that it is done quickly and sometimes at lower costs [Qui94].

Distributed system provides a seamless integration of computing functions between different computers [FG07]. Distributed Computing is one in which components located at networked computers communicate and coordinate their actions only by passing the messages [GCK01]. The major goal of constructing the distributed systems is to obtain the large-scale resource sharing at affordable cost. Examples of distributed systems include: Internet, Intranet, Mobile Computing and Ubiquitous Computing.

According to San et al. traditionally, there have been three categories of "Distributed Computing" [JP03]:

- Cluster Computing: Similar machines are joined together to form a cluster of virtual machines known as cluster computing. Linux clusters are good examples of cluster computing.

- Peer-to-Peer Computing: In Peer-to-Peer Computing many desktop computers are linked together to aggregate the processing power. The distinguishing characteristic is the desktop computers itself, which almost exclusively is a low-power client PC.

- Grid Computing: It connects a wide variety of computer and other computing resources such as printers, desktops, laptops, databases, storage area networks *etc.* to create vast "virtual" reservoirs of computers serving geographically widely separated users.

  Detail distinction between all these is presented in Table 1.1. Grid Computing allows heterogeneous resources to be connected and is therefore quite generic in nature. The focus of this work is mainly on Grid Computing, which is explained in following section.

## 1.1 Grid Computing

Focusing upon the efficient attributes of Grid Computing this section elaborates the past, present and future trends of Grid Computing. The term Grid Computing was conceived in the mid- 1990s to denote a new infrastructure of distributed computing for the scientists and engineers in a more advanced scope. This name was inspired by the electrical power energy because of its pervasiveness, ease of use and reliability [FK05]. Grid Computing provides a virtual computing resource that will be used to execute applications in a Virtual Organization (VOs) [BJ03]. VOs can span from small corporate departments that are in the same physical location, to large groups of people from different organizations that are spread across the globe. VOs can be large or small; static or dynamic in nature. Some examples of VOs are an accounting department of a company, an emergency response team *etc.* [Abb04].

Grid Computing is the generic term given to techniques and technologies designed to make pools of distributed computing resources available on-demand. More generic definition of Grid is [IFT02]:

Table 1.1: Distinction between Cluster, Peer-to-Peer and Grid Computing

| ***Characteristics*** | ***Cluster*** | ***Peer-to-Peer*** | ***Grid*** |
|---|---|---|---|
| **Target Communities** | Very less professional communities | It has been popularized by file-sharing and highly parallel computing applications | Motivated by Professional Communities |
| **Evolved In** | With in the Local Area Network | Harnessing the computing storage and communication power of the nodes | Coordinating and problem solving in VOs for industries and businesses |
| **Resource Management** | Centralized | Distributed | Distributed |
| **Resource Ownership** | Singular (Often locked to a single node to prevent data corruption) | Singular or multiple, varies from platform to platform | Singular, multiple, or distributed, depending on the architecture |
| **Method of Resource Allocation / Scheduling** | Centralized, allocated according configuration | Decentralized | there is no single permanent host for centralized data. Everything is transient. |
| **Discovery Mechanism** | Defined membership (Static or Dynamic) | Centralized index, as well as, multiple decentralized mechanisms | Always decentralized discovery mechanism. |
| **Inter-Operability** | Guaranteed within a cluster | Enforced within a framework | Multiple competing standards |
| **External Representation** | Single Image | Single or multiple image(s) Unknown | it is circumstantial Inter-Operability |

| | | | |
|---|---|---|---|
| **Suggested Equipment** | Mostly high-end, high capability systems | High-end or commodity systems | Any type, including wireless device and embedded systems |
| **Virtual Organization (VO)** | As deals with singlr organization so, VO, | VO with generis Resource Sharing | VO with service semantics |
| **Quality of Service(QoS)** | QoS is very less | QoS is less | QoS is High |

*"A Grid is a hardware and a software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."*

Today, Grid is attracting high number of professionals and huge investment from various funding agencies because its future is potentially revolutionary. It has been developed extensively in recently years and has become an important platform for high performance computing in scientific areas [LPN04].

A well-known example of Grid Computing in the public domain is the ongoing SETI (Search for Extraterrestrial Intelligence) @Home project in which thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of "rational" signals from outer space [SET08]. Perhaps the most ambitious is Oxford University's Centre for Computational Drug Discovery's project that utilizes more than one million PCs to look for a cancer cure. People around the world donate a few CPU cycles from their PCs through "screensaver time." The project eventually will analyze 3.5 billion molecules for cancer-fighting potential *etc.* [Cor08].

Grid Computing is finding its applications in various application domains. At the same time it is also having a promising future due to the following reasons:

- Grid is a cluster of clusters and gathers the unused resources.
- Grid Computing provides high performance per workstation.
- It works for the distributed users, large scale and distributed resources, across domains and faster interconnected network.
- It has the ability to make more cost-effective use of a given amount of computer resources.
- It suggests that the heterogeneous resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as collaboration toward a common objective.
- It solves the problems that can not be approached without an enormous amount of computing power.
- It provides an approachable interface for managing and access to the resources. This may includes scheduling policies, licence checking softwares, access rights and security privileges *etc.*
- Grid Computing develop and deploy the optimal applications on the distributed environment to take the advantage of heterogeneous resources.

### 1.1.1 Characteristics of Grid Computing

The purpose of the Grid Computing is to enable a new generation of applications, combining intellectual and physical resources that span many disciplines and organizations. It also provides more effective solutions to important scientific, engineering, business, and government problems. In the following paragraph ten important characteristics has been discussed namely, Heterogeneity, Geographical Distributed, Resource Sharing, Multiple Administration, Scalability, Dynamistic or Adaptability,

Uncertainty, Demandable Access, Consistent Access and Pervasive Access of Grid Computing.

- **Heterogeneity:** One of the most important characteristics in Grid environment is heterogeneity. It aggregates large numbers of independent and geographically distributed computational and information resources, including supercomputers, different clusters, network elements, data storages, sensors, services, and Internet networks.
  - Resources are heterogeneous.
  - Resources are administratively and geographically disparate.
  - Users do not have to worry about system details (*e.g.* location, operating system, accounts *etc.*).
  - Resources have different resource management policies.
  - Resources are numerous that are owned and managed by different, potentially mutually distrustful, organizations and individuals that are likely to have different security policies and practices.
- **Geographically Distributed:** In Grid Computing resources may be located at different geographical locations and are loosely coupled.
- **Resource Sharing:** Resources in a Grid Computing belongs to many different organizations and geographical distributed that allow other organizations (*i.e.* users) to access them. Nonlocal resources can thus be used by applications promoting efficiency and reducing cost.
- **Multiple Administrations:** Each organization may establish different security and administrative policies under which their owned resources can be accessed and used.

- **Scalability:** Grid may be able to grow from few resources to millions. This raises the problem of potential performance degradation. Consequently, applications that require a large number of geographically located resources must be designed to be extremely latency tolerant. For example a Grid should operate equally well with 100,000 participants as with 100.

- **Dynamistic or Adaptability:** In a Grid Computing, a resource failure is the rule, not the exception. In fact, with heterogeneous nature of resources, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behavior dynamically so as to extract the maximum performance from the available resources and services.

- **Uncertainty:** Uncertainty in Grid environment is caused by multiple factors, including

  - *Dynamism*, which introduces unpredictable and changing behaviors that can be detected and resolved only at runtime.
  - *Failures*, which have an increasing probability of occurrence and frequencies as system/application scales increase.
  - *Incomplete knowledge of global system state*, which is intrinsic to large decentralized and asynchronous distributed environments.

- **Dependable Access:** Grid Computing must assure the delivery of services under established Quality of Service (QoS) requirements. The need for dependable service is fundamental since users require assurances that they will receive predictable, sustained and often high level performance.

- **Consistent Access:** Grid Computing must be built with standard services, protocols and interfaces thus hiding the heterogeneity of the resources while allowing its scalability. Without such standards, application development and pervasive use would not be possible [MLBLGS03].

- **Pervasive Access:** Grid Computing must grant access to available resources by adapting to a dynamic environment in which resource failure is commonplace. This does not imply that resources are everywhere or universally available but Grid Computing have its different policies and standards to extract the maximum performance from the available resources.

Grid Computing is often regarded as a means for creating inexpensive *super computing*. The purpose is to enable inexpensive, scalable and reliable computing, usually across corporate boundaries [MIH04]. All the aforementioned characteristics of the Grid Computing can be understood in depth by studying its layered architecture.

## 1.2 Grid Architecture

The Grid Architecture proposed by Ian Foster, Carl kesselman and Steven Tuecke is layered. It is based on the open standards, and connectivity protocols, which facilitates the sharing of resources. Figure 1.1 illustrates the layer of the architecture with specific capability at each layer [IF01a]. Different layers of Grid Computing Architecture are:

- Fabric Layer: Interface layer to resources

  The Fabric layer defines the protocols, interfaces and resources that can be shared. This includes the computational resources, data storage resources, network resources, different clusters, and other system resources. These resources can be physical (*e.g.* computational and network resources, storage systems, sensors) or logical (*e.g.* distributed file systems, different clusters, & geographically distributed systems) by nature. Both the resources implement the inquiry mechanism that permit discovery to their structure, state and capabilities. This layer provides the logical view of the resources rather than a physical view [IF01a]. For example the view of a cluster with a local resource manager is defined by local resource manager and not the cluster hardware.

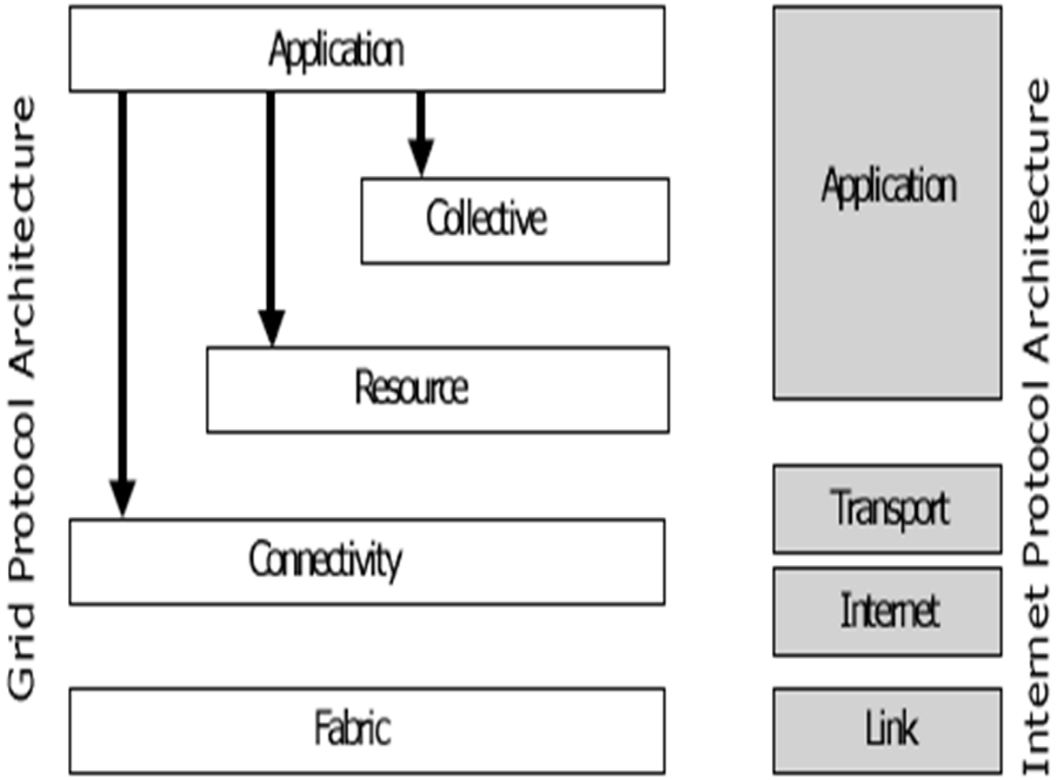



FIGURE 1.1: The layered Grid architecture and its relationship to the Internet protocol architecture

- Connectivity Layer: Communication Easily and Securely

  Connectivity layer defines the core communication and authentication protocols that are required for the Grid specific network transactions. Communication protocols are: internet protocols such as TCP/IP internet protocol stack, DNS, ICMP, UDP and various routing protocols. Communication protocols enable the exchange of data between fabric layer resources. Authentication protocols build on communication services provide cryptographically secure mechanisms for verifying the identity of users and resources. They must rely on existing standards *e.g.* X.509, TLS, SSL, and Kerberos.

- Resource Layer: Sharing of Resources

The resource layer builds on the communication protocols and authentication protocols to define protocols for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. This layer only handles the individual resources and hence, ignores the global state and atomic actions across other resource collection (which in the operational context is the responsibility of the collective layer). Protocols defined at this layers includes management and information protocols like:

  - Grid Resource Allocation Manager (GRAM ): It includes remote allocation, reservation, monitoring, and control of resources.
  - Grid File Transfer Protocol (GridFTP): It includes high performance data access and transport.
  - Grid Resource Information Service (GRIS): It includes access to structure and state information.

  All these protocols are built on the connectivity layer and utilize standard Internet Protocols (IP) for communications.

- The Collective Layer: Coordination of multiple resources
  As resource layer manage the individual resources, the collective layer is responsible for all global Resource Management and interaction with the collection of resources. The collective layer defines various protocols that provide the system-oriented capabilities for wide scale deployment. Collective functions can be implemented as persistent services with associated protocols. Some key examples of collective services are: discovery services, collaborative services, monitoring and diagnostic services, coallocation, scheduling and brokering services, software discovery services, data replication services, community accounting and payment services, Grid enabled programming services *etc.*

- Application Layer: User Defined Application Layer

  This layer defines the protocols and the services that are targeted towards a specific application or a class of applications at each lower layer. The application(s) can directly access the resource, or can access the resources through collective service interface APIs. Applications are constructed in terms of, and by calling upon, services defined at any layer.

As the introduction of the Grid Computing clearly shows the history, future trends, different layers of the architecture, resources and processes, to manage the proper functionality of all these we need an important component *i.e.* Resource Management. Therefore, it is necessary to understand the concept of resource management.

## 1.3 Resource Management

Resource Management consists of two terms *i.e.* Resource 'plus' Management. Resource, in terms of organization, is name of Assets, Men, Machine and Materials. Management is selection and implementation of best possible alternative to use of the available resources in effective and efficient manner in order to attain the desired goals. Hence, Resource Management is the management of resources directed towards the desired set of outcomes. It is an efficient and effective deployment of an organization's resources as and when they are needed. Resources are differing by:

- Type (storage, calculation, sensor, software *etc.*)
- Platform (OS, Architecture *etc.*)
- User (human skills)
- Owner (financial resources, inventory, production resources, or information technology (IT))

- Hardware (PC, cluster, smart phone *etc.*)
- Location (with in the department, with in the campus *etc.*)

Resource Management is the central and integral component of Grid Computing. Resources are the entity that is to be shared, which may be physical or logically distributed in the Grid Computing. Management is the process of how resources are exposed and made available for use on Grid. And Resource Management is function to control the resources, made available to others.

The overall goal of the Resource Management is to efficiently discover and schedule the applications that need to be utilized the available resources in heterogeneous and dynamic environments. Grid Resource Management is defined as a process that includes following:

- Resource Requirement Identification: The first and foremost component of the Resource Management is the *resource requirement identification*: A Resource Specification Language (RSL) has been developed as a common notation to exchange of the information between the applications and local resource managers [Abb04]. RSL provides the two types of information: Resource Requirements and Job Configuration:

  **Resource Requirements:** It contains information about the machine type, number of nodes, memory, *etc.*

  **Job Configuration:** It containes information about the directory, executable, arguments, environment *etc.*

- Resource Discovery: Discovery allows the user to discover, resources as services with the requisite properties and functionalities. The problem of Resource Discovery is that given an application with a requirement specification of the types of computing resources, locate the machines out of the available machines that meet the specified criteria. This is the process of querying the distributed

state of the Grid to identify those resources whose characteristics and state match those desired by the resource consumer [FK05]. The detailed description of resource discovery process has been explained in Section 1.4.

- Resource Matchmaking: Matchmaking is a process of finding the list of resources that satisfies the requester's request. Matchmaking is executed based on whether the resource request and the resource advertisement match or not. The match between requests and advertisements is made based on the match between capabilities of the resource available and resource requested [VS07]. The major descriptive language that used for matchmaking is classified ads (ClassAds). The various components of matchmaking as defined by Horst Wenske are: ClassAds Specifications, Advertising Protocols, Matchmaking Algorithms, Matchmaking Protocols, and Claiming Protocols [Wen04].

- Scheduling and Monitoring:
  Grid Scheduling is the process of arrangement of a number of related operations *i.e.* resources in accurate time, based upon different policies and standards. List of Grid Scheduling policies for malleable applications are: when jobs are available, when a failure occurs, when a job is waiting *etc.* as discussed in [BE07].

  - When a job is waiting for placement. Two cases may occur. If enough resources are available, then the job is placed. Otherwise, the job is queued (favoring executing jobs), or shrink messages are sent to make room for the job (favoring pending jobs).

  - When some resources are available. These resources are distributed over executing malleable jobs and pending jobs. Depending on the order in which they are considered, jobs in one category are favored over jobs in the other one.

  - When a job requests for growing. If enough resources are available, growth is granted. Otherwise the scheduler attempts to shrink other malleable jobs before denying the growth.
  - When a failure occurs. The scheduler let applications handle failures on their own. The scheduler gives no guarantee on the quality of the resources.

  Similar to scheduling, monitoring is another important issue in Resource Management. A wide variety of processing tasks may be using the resources of the Grid, and these needs to be monitored in real-time and both normal and abnormal events must be reported using various modalities. Grid environment provides a mechanism for real time monitoring and notification. For real time monitoring like type of events: categorization can be done as minor, moderate, serious or urgent. Where as notification policy may require notification via the most intrusive means (say, ringing a pager or telephone) for serious or urgent events, e-mail notification for moderate events, and no notification for minor events [Abb04].

- Policy and Security in Resource Agreements: The purpose of resource policy is to control *by whom and how* the resources may be used. For example, the resource may be used only by certain users or during specific times of the day or may require different level of payments for different services. Thus, the policy will govern the Service Level Agreements (SLA) to which the resource provider is willing to agree. Policy enforcement is symmetric: it may be applied by both entities entering into an SLA. Resource agreements can encompass not only a commitment to perform a task but also commitments to the level of performance, or quality of service. While Security is the priority concern in any computer system, same for the distributed system such as Grid. According to Ramakrishnan, Grid Security is a multidimensional problem [Ram04]. An

effective security policy mechanism, or a solution, should be such that it will: establish the identity of the user *i.e.* authenticate and track the authentic user's actions in the Grid Computing [Pra08].

- Resource Allocation: Resource Allocation means to provide the Grid services to the user or customer including applications, resources *etc.* Allocation often appears in context of virtualization, orchestration, utility computing, open configuration concepts and projects. Resource Allocation for applications has several advantages over best effort execution.

  - Firstly, it allows the user or workflow manager [ea05] to control the scheduling and execution of the application on the allocated resources.
  - Second, it allows the user or workflow manager to explore the space of resources to be allocated and optimize for performance without worrying about external factors such as the workload of the resource or the policies of the resource provider.
  - Third, it enables the adaptation of the application based on the allocated resources in order to achieve a certain level of performance.
  - Fourth, it allows the execution of applications that require co-allocation of resources or have other constraints on the resource requirements.

- Problem Determination and Fault Management: As Grid Computing is a distributed system involving heterogeneous resources located in different geographical domains, for this case fault tolerant and resource allocation services have to be provided [Sig05]. In particular, when crashes occur, tasks have to be reallocated quickly and automatically, in a completely transparent way from the users point of view. Various methods such as checkpointing is required for managing the abrupt abortions or job terminations, so that restart or recovery can be easily achieved, thus improving the system availability and reliability.

Resource Management scenarios often include resource discovery, resource monitoring, resource inventories, resource provisioning, fault isolation, variety of autonomic capabilities and service level management activities. The promise of Grid Computing, for effectively harnessing computing resources across the organization, is enormous. Resource Agent, a core part of Grid system, heads the whole hierarchy of maintaining all the service information of the system. Agent performs various processes like, discovering, scheduling, allocation and evaluation. Out of these resource discovery is first and foremost important process.

## 1.4 Resource Discovery

As described in the previous section, Grid environment consists of large number of shared resources distributed among the many locations. Discovery is the core component of the Resource Management *i.e.* performed on the distributed resources, applications *etc.* Discovery is an information-gathering process, meant to dig deep into the details of what is important to a client's business, target audience, and industry. Resource Discovery is a very important process because it allows you to see when and where things happened and to collect the information necessary to assess against your desire. Resource Discovery can be defined as a directory service directed to the spontaneous networks environment [GC01]. Discovery in the Grid environment becomes complex as the resources are geographically distributed, heterogeneous in nature and owned by different individuals and organizations each having their own resource management policies and different access and cost models.

The Grid Resource Discovery problem can be defined as the problem of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements [VI04]. Resource Discovery is performed by Resource Management system to obtain information about the available

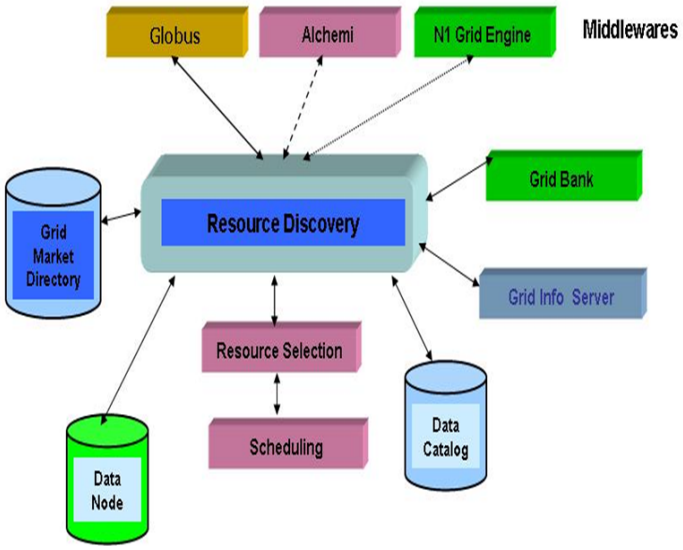



FIGURE 1.2: Resource Discovery Process

resources. Resources can enter or leave at any time in Grid Computing environment, Resource Discovery mechanism provides the information about the available resources in a specific moment [RdM06]. Resource Discovery deals with finding, locating, searching and organizing the matching data (information, facts, quires, numbers *etc.*), in response to queries of users or an automated mechanism with following trades off:-

- In Shortest amount of time
- With special access policy and standards
- At minimum cost

Resource Discovery systems become more and more important as distributed systems grow and as their pool of resources becomes more variable. According to Koen et al. Resource Discovery systems can be categorized by different design aspects mentioned in [KV05]. Each of these aspects represents a design choice for which several solutions are possible. The design aspects are: service provider, construction, foreknowledge, architecture, registration, discovery, supported resources,

naming and queries.

Resource Discovery has generally two key steps [ICJ02]. The first step is to **locate** a resource, and the second step is to **connect** or **match** to it. Locating resource means getting a query to metadata or data. Resource locating would not be quite difficult in peer-to-peer network that is managed by a central server. Traditional lookup service, *i.e.* LDAP (Light Weight Directory Access Protocol) service, maintains all information, such as name, location, attribute in a single server that is normally based on central DBMS (Database Management Systems), which will result constant search performance. But it becomes variable when arbitrary joins and leaves are allowed. Where as matching means determine a match between query and metadata.

Unfortunately, Grid Computing technology is still far away from reach of inexperienced application users, *e.g.* computational scientists and engineers. Resource Discovery is a challenging problem in Grid Computing because computational resources are large-scale geographically distributed. In this thesis, Resource Discovery algorithms has been designed and identifying the specific QoS parameters under which it can perform efficiently. The performance of algorithms underneath TU-Grid Testbed for varying network conditions based on various evaluation criterions such as Time, Cost, Reliability and Security have been taken. The next chapter describes the various components, models, approaches and algorithms that presents the existing solutions for the Resource Discovery. A survey of all those methods has also been presented.

## 1.5 Thesis Organization

This section discusses the outline made in the thesis are structured given below:

**Chapter 2:** *Literature Review*

This chapter discusses the background and related work done so far in the area of Resource Discovery in Grid computing. This chapter is divided into three parts: in first part, Resource Discovery process, its components and characteristics are explained. In second part, Resource Discovery models, approaches, and algorithms are explained. In third and last part a comprehensive review of the work done in Resource Discovery is reported. Based upon the literature review gaps have been identified where improvements are needed. This gap analysis gives us a basis for problem formulation of this thesis.

**Chapter 3:** *Identification and Analysis of Quality of Service Parameters*

This chapter starts by identifying the key design goals that need to be considered for design and implementation of Resource Discovery algorithms. The Quality of Service (QoS) parameters, dynamic decision factors and Service Level Aggrement (SLA) for the efficient Resource Discovery process in the Grid Computing have been discussed. This chapter also discusses various modules of QoS used by SLA and its generic interface by using the campus wide case study. QoS is the capability of a network to provide better services to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks. QoS deals with generic application needs timeliness, reliability, and security. QoS mechanisms in Grid computing require to support more then the guaranteed availability of given resources. QoS guarantees in such a context not only the behavior of the single components of the Grid system, *i.e.* processor availability on

a cluster computer, but also defines, how the whole Grid environment manages the given job. To enhance the efficiency of Resource Discovery with QoS, a case study of campus Grid has also been presented in this chapter.

**Chapter 4:** *Proposed Resource Discovery Algorithms*

This chapter presents proposed Resource Discovery algorithms and their implementation details. Three Resource Discovery algorithms have been proposed, designed and developed based on different assumptions and scenarios to achieve efficient resource discovery in Grid environments. These algorithms address and improve upon some parameters in the existing algorithms like Time, Cost, Reliability and Security. At the core of the implementation strategies, a Resource Discovery testbed is used to decide what, when and where the algorithms are useful. All the algorithms have been implemented and tested for their efficiency.

**Chapter 5:** *Deployment, Testing and Validation of proposed Algorithm*

The TUGrid tesbed is initially designed and implemented to test the Resource Discovery process is explained in Section 1 of this chapter. Next section explains the evaluation parameters of three algorithms and last section presents various experiments that evaluate the designed algorithms with the functionalities of TUGrid testbed. Different types of QoS parameters including time, costs, matching scenarios, reliability and security.

**Chapter 6:** *Conclusion and Future Scope of the work*

This chapter concludes the research work carried out in this thesis. In support to thesis work:

A) different QoS parameters have been analyzed (like time, cost, reliability, security) etc.

B) Three algorithms have been proposed namely: Path optimization Algorithm, Weight Optimization Algorithm and Request Matching Algorithm. These algorithms have been designed and developed using the different approaches and different assumptions. All the three aforementioned algorithms have been deployed and tested on the TUGrid testbed. The efficiency of the algorithms have been successfully demonstrated by taking the different QoS parameters under considerations.

# 2

# Literature Review

*Grid Computing is the next generation of IT infrastructure. It consists of collection of heterogeneous resources spread across multiple administrative domains. Grid Computing provides scalable, secure and extremely high performance mechanism for discovery and access to remote computing resources, in a seamless manner [JF04]. Grid Computing has the potential to become the main execution platform for high performance and distributed applications. However, such systems are extremely complex. The possibility to discover an enormous amount of resources to a parallel application (thousands of machines connected through the Internet) and to make it with lower cost than traditional alternatives (based on parallel supercomputers) is one of the main attraction and big challenge in Grid Computing. In order to cope*

*with the dynamic nature of Grid Computing, some way of incorporating dynamic state information about the available resources into its decision-making process is needed. This process is termed as Resource Discovery. Many projects have been designed and implemented to incorporate Resource Discovery process, with a variety of models, approaches and algorithms [AA04]. This chapter focuses mainly upon the existing research in the area of Grid Resource Discovery. Organization of the chapter is as follows: Section 2.1 discusses the Resource Discovery: core of the Grid Computing. Section 2.2 surveys the existing Resource Discovery Models, Approaches and Algorithms. Section 2.3 captures the actual experience regarding the Resource Discovery in different middleware. Section 2.4 discusses the efficiency in the Resource Discovery process. Section 2.5 presents the problem formulation of the thesis. Finally, Section 2.6 presents the objectives of the thesis.*

## 2.1 Resource Discovery: Core of Grid Computing

In the context of Resource Discovery, Grid is a collection of geographically distributed nodes, that may join or leave at any time without notice (due to communication failures) [Iam03]. The Grid Resource Discovery problem can be defined as, the problem of matching an application's requirement for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements. The problem is complicated by the fact that some resource information (*e.g.*, CPU load or available storage) change dynamically. The problem may be viewed as one of the efficient access to widely distributed real-time data streams representing the state of Grid resources [VI04]. Resource Discovery is a basic service in Grid Computing, which gives the description of resources desired and finds the available ones to match the description. In Grid, how to discover resources efficiently, has become a crucial factor to evaluate the performance of the whole system [HZS05]. The problem of the Resource Discovery is studied in detailed in

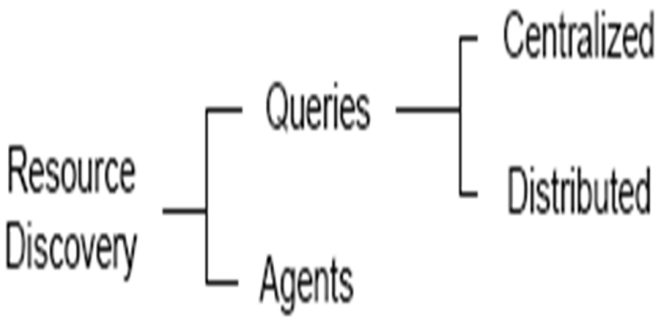



Figure 2.1: Resource Discovery Taxonomy [KKM01]

the following sections.

### 2.1.1 Resource Discovery Taxonomy

Major function of Resource Management in Grid Computing is to provide a mechanism for resources in the Grid to be discovered and utilized by network application. Resource Discovery provides matching resources. Discovery is initiated by a network application to find suitable resources within the Grid. Figure 2.1 shows the taxonomy for Resource Discovery process [KKM01]. Query based system are further characterized depending on whether the query is executed against a distributed database or a centralized database. Agent based approaches, send active code fragments across machines in the Grid that are interpreted locally on each machine. The major difference between a query based approach and an agent based approach is explained in the Section 2.2.2.

### 2.1.2 Resource Discovery Components

There are numerous components of the Resource Discovery that are used to dig up the new solutions for the Resource Discovery process. According to Iamnitchi [Iam03] four-solution space for the Resource Discovery are:

- Membership Protocol: The membership management protocol is responsible for incorporating new members or nodes into the system and reallocates members from group that has too few members.

- Overlay Construction: Overlay Construction functions select the set of active collaborators from the local membership list. In practice, this set may be limited by factors such as available bandwidth, message-processing load, security or administrative policies, and topology specifications [MHBL99].

- Preprocessing: Preprocessing refers to the off-line preparations for better search performance, independent of request. A very well known example of preprocessing technique is dissemination of resource descriptions that is, advertising descriptions of the local resources to other areas of the network for better search performance and reliability.

- Request Processing: The request processing has further two components:
  **Local processing:** Looking up the requested resource in the local information. (*e.g.* a request for A and B could be broken into two distinct requests to be treated separately), and/or applies local policies, such as dropping requests unacceptable for the local administration. The local information is maintained by the every node using the above-mentioned preprocessing techniques.
  **Remote processing**: The remote component implements the request distribution rule: sending requests to other peers through various mechanisms (flooding, forwarding, and epidemic communication).

## 2.2 Existing Solutions for Resource Discovery

Starting from the Mor Harchol et al. [MHBL99] till now there have been number of solutions for the Resource Discovery *i.e.* Resource Discovery Models, Resource Discovery Approaches and Resource Discovery Algorithms. All are presented in

details in the coming sections.

### 2.2.1 Resource Discovery Models

Different attempt to categorize the heterogeneous models for the Grid Computing and performance metrics are discussed in [SB05]. Grid environment uses the three models: a pull model, a push model and a push-pull model for Resource Discovery [ML05]. The major function of these models is used to update the resource status database described as follows.

**The pull model:** In this model, a single daemon associated with the scheduler can query Grid resources and collect state information such as CPU loads or the available memory. The pull model for gathering resource information incurs relatively small communication overhead, but unless it requests resource information frequently, it tends to provide fairly stale information, which is likely to be constantly out-of-date, and potentially misleading. In centralized scheduling, the Resource Discovery could be rather interfering and begin to take significant amounts of time as the environment being monitored gets larger and larger. The architecture of the pull model is shown in Figure 2.2.

**The push model:** In this model, each resource in the environment has a daemon for gathering local state information, which will be sent to a centralized scheduler that maintains a database to record each resource activity. If the updates are frequent, a precise view of the system state can be maintained over time; obviously, frequent updates to the database are interfering and consumes network bandwidth. Figure 2.3 shows the architecture of the push model.

**The push-pull model:** The push-pull model lies somewhere between the pull model and the push model. Each resource in the environment runs a daemon that

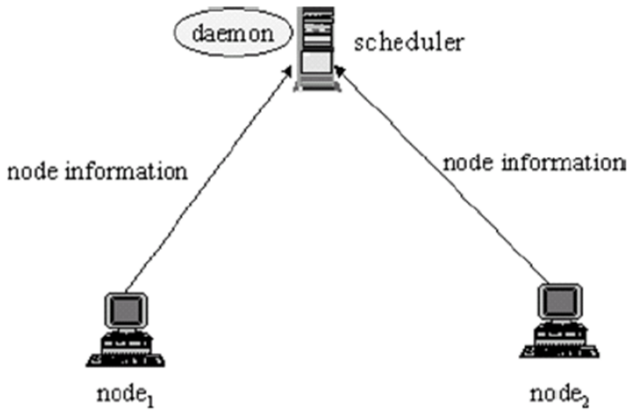



FIGURE 2.2: Pull Model for Resource Discovery [ML05]

collects state information. Instead of directly sending the information to a central scheduler, there exist some intermediate nodes running daemons that aggregate state information from different sub-resources that respond to queries from the scheduler. The architecture of push-pull model is shown in Figure 2.4.

### 2.2.2 Resource Discovery Approaches

There are various approaches that exist for Resource Discovery in Grid environments. Broadly can be classified as *Query Based* and *Agent Based*. In a query based approach the resource information is queried for resource availability. Most of the contemporary Grid systems follow this approach [Kan03]. Query based system are further characterized depending on whether the query is executed against a distributed database or a centralized database.

An agent is a software entity permitting to achieve tasks which bridges the gap between Grid application users and Grid resources. It is autonomous, reactive and capable to communicate with others [DB05a]. Each agent is an independent entity, with its own structure, goals and resources; but an agent must communicate with

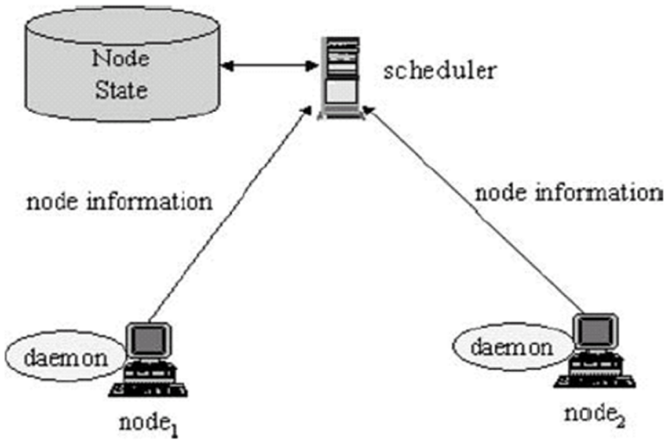



Figure 2.3: Push Model for Resource Discovery [ML05]

other agents in its environment to meet its goals [DB05b], [FBW02], [QC02], [Sie01]. One important feature for agents is that they are processed stage by stage between neighboring agents, to adjust their advertisement and discovery behaviors; thus adapting to the highly dynamic Grid environment [SD05]. This feature plays an important part in system scalability.

An agent-based Grid service discovery has been presented by Kyungkoo et al. and Li Changln et al. in [LC03], [KJM00]. Agent based approaches send active code fragments across machines in the Grid that are interpreted locally on each machine. Furthermore, agents can also passively monitor and can distribute resource information either periodically or in response to another agent. Thus, agents can mimic a query based Resource Discovery scheme. According to Rajkumar Buyaa et al. agent based Resource Discovery is inherently distributed [KKM01]. The major difference between a query based approach and an agent based approach is that agent based systems allow the agent to control the query process and make Resource Discovery decisions based on its own internal logic rather than residing upon a fixed function query engine. Most agent systems are based on an underlying mobile code environment like Java.

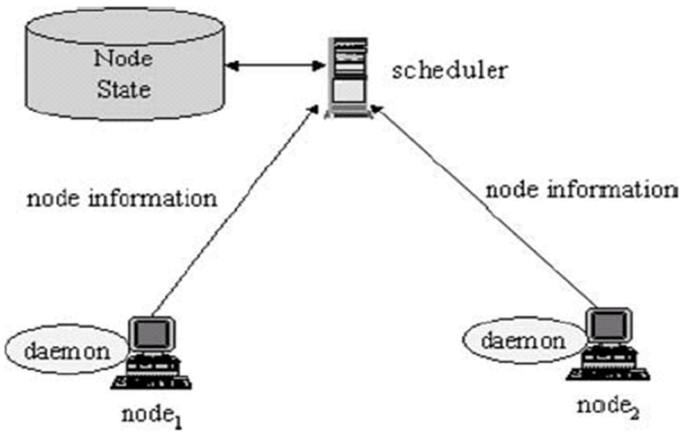



FIGURE 2.4: Push-Pull Model for Resource Discovery [ML05]

### Peer-to-Peer Approach

The peer-to-peer (P2P) paradigm is based on the principle that every component of the system has the same responsibilities acting simultaneously as a client and as a server, as opposed to traditional client-server model [ea06b]. In P2P approach Resource Discovery is centralized. Iamnitchi proposed P2P Resource Discovery architecture for a large collection of resources in [IF01b]. Different Resource Discovery problems are there in large distributed resource-sharing environment especially in Grid environment [Iam02]. Four different architectural components and four environment parameter factors are identified, which dominate the performance and design strategies for a Resource Discovery solution. Iamnitchi also gave the four axes framework, it is possible to design any Resource Discovery architecture in a Grid Computing [Hos02].

### Ontology Description- Based Approach

Ontology Approach is flexible and extensible for performance of the Grid resource selection mechanism [SAL02]. Ontology refers to a description of a services (resources). The main idea behind this approach is to advertise the services (resources). Since,

in Computer Science, Ontology has been used for several years to build a vocabulary of a specific domain application area. Ontology can be distributed and shared for utilization [AP05]. However, it provides a clear understanding of characteristics and properties of classes and relations.

Ontology has various components that are described in [Gru93]. In this approach, service provider registers its service description into the service registry database. When a Grid application sends a request to service directory, matchmaker returns the matches to the service requester. Requester chooses the best resource based on the specific need. A semantic service matchmaking framework in a Grid environment has been proposed which claimed that this matchmaking framework can provide a better service discovery and also can provide close matches [SL02]. Bilateral and multilateral matchmaking (*i.e.* gangmatching) services are discussed in detail by Raman in [Ram01]. Supporting to this approach, different algorithms also exist which were suggested by CMU [MPS02], [mfws], TRASTOUR [DTGC01], and LARKS [KSL99], [SSWL02]. This approach has been used in one of the proposed algorithm *i.e.* "Request Matching Algorithm".

**Routing Transferring Model - Based Approach**

A Resource Discovery technique called Routing Transferring Model is proposed by Wei Li et al. in [WL02]. This model consists of three basic components: resource requester, resource router and resource provider. The provider sends the resource information to a router and router stores the information in a router table. After that, when requester sends a request to the router, router checks its routing table for an appropriate resource provider. After finding that entry router forwards that request to the service provider or another router. The authors formalized the model and analyzed the complexity of Shortest Distance Routing Transferring (SD-RT) algorithm based on the formalization. Authors also claimed

that Resource Discovery time depends on the topology and they also showed that SD-RT could locate a resource in the shortest time, if the topology and distribution of resources are explicit. They examined their proposed model in Vega Grid project [NS02], [ZX02b], [ZX02c], [ZX02a], [Zhu02] and their experiment shows that higher frequency and more location of resources can reduce the Resource Discovery time.

**Parameter - Based Approach**

Different approaches for Resource Discovery in a Grid system are described by Maheswaran in [MK00]. A new concept "Grid potential" and "Data Dissemination Algorithm" were proposed, which encapsulates the processing capabilities of different resources in a large network. The proposed algorithm follows swamping approach for message distribution [MHBL99]. When a message comes to a node, that message gets validated. The validation process depends on three types of dissemination; universal awareness, neighborhood awareness, and distinctive awareness.
In *Universal Awareness,* the information such as node can learn about every other nodes in the Grid. Consequently, it causes significant amount of communication in large network sizes for message transfers. In *Neighborhood Awareness,* the information such that a node learns about the other nodes that are less than a fixed distance away from it. In *Distinctive Awareness,* the term Grid Potential is used to determine the distinctive awareness.

**Request Forwarding Approach**

According to Iamnitchi following four-request forwarding Approaches are identified [Iam03].
a.) *Random Walk Approach:* In this approach, to forward the request, the node is chosen randomly.
b.) *Learning-Based Approach:* A request is forwarded to a node who answered

similar request before. If no similar answer is found, the request is forwarded to a randomly chosen node.

c.) *Best-Neighbor Approach:* The number of received answer is recorded without recording the type of requests. The request is forwarded to that node which answered highest number of requests.

d.) *Learning-Based + Best-Neighbor Approach:* This Approach is identical to learning-based Approach except when no similar answer is found. Request is forwarded to the best neighbor.

### Quality of Service (QoS)-based Approach

An algorithm to discover the occasionally available resources in a multimedia environment has been proposed by Huang et al. in [YH02]. Different policies for QoS based Resource Discovery service for a given graph theoretic approaches are proposed by Huang et al. A generalized version of Discovering Intermittently Available Resources (DIAR) algorithm based on occasionally available resources is introduced [MHBL99]. The performance of QoS policies based on different time-map strategies in a centralized system is evaluated [YH02]. According to that experiment, it is found that randomized placement strategies and increased server storage can facilitate better performance to discover a particular resource.

### Comparative Analysis of all Approaches

Comparison between the Resource Discovery Approaches based upon various features including scalability, reliability, adaptability and manageability has been done and placed in Table 2.1 [SB08]. Large numbers of parameters are to be taken into account for dealing with complexity of Grid Resource Discovery as it becomes complex with the increase in size of Grid. For dealing with the large global Grids, P2P is the best approach to be used, as it uses the graph theoretic Approach to achieve scalability and manageability. De-Centralized Resource Discovery approach can also

be used for large Grid, but it is limited to certain extent because of its managing units and dependency on other approaches. However, Ontology based approach uses the central broker for matchmaking, which reduces its scalability.

In parameter based approach a new concept of Grid Potential is introduced, which helps to achieve the efficient way for maintaining the current status of resources. On the other hand QoS approach helps the clients with the reliable feedback about the expected behavior of the system as a whole. Request Forwarding approach is not much used today because of its nonstandard nature. Comparison among the different approaches reveals that Routing Transferring approach is the fastest way to discover the resources for small Grid as it uses the shortest distance routing table for matchmaking process.

### 2.2.3 Resource Discovery Algorithms

Number of algorithms has been developed for Resource Discovery process in Grid Computing. The work was started by Mor Harchol et al. with Resource Discovery in distributed networks [MHBL99]. In their work, they proposed the algorithm and named it "Name Dropper", which is efficient in terms of time and communication required. Following it A. Aziz and V. Garg proposed an algorithm named "Deterministic Algorithm for the Resource Discovery problem" [VG00]. An algorithm named "Absorption" algorithm has been proposed by C. Law in [CL00]. In their dissertation the performance measures were time complexity, message complexity and pointers complexity. A Fast self- stabilizing algorithm from a directed graph with out-degree 1 for the overlay networks has been developed in [JA07]. Ittai and Danny worked with "Asynchronous Resource Discovery" and also, proposed an algorithm named generic algorithm in [IA06]. It is used for the upper and lower bounds for the Resource Discovery problem.

Table 2.1: Qualitative Comparison of All Approaches

| *Approach/ Property* | *P2P* | *Ontology* | *Routing Transferring* | *Parameter* | *QoS* | *Request Forwarding* |
|---|---|---|---|---|---|---|
| *Base Approach* | Agent/ Query | Agent | Query | Agent | Query | Agent/Query |
| *Scalability* | High Scalability as it uses the four axes framework | Less Scalable as Broker is centralized | Scalable as it uses the routing protocol | Scalable due to Grid potential concept used by it | Uses different tie map strategies in centralized system to increase scalability | Nodes are chosen randomly which make it scalable |
| *Reliability* | Based on graph theoretic hence reliability increases | Failures are detected as soon as they occurs so more reliable | Quite reliable as it uses the routing concept | Reliable as nodes can add and deleted from anywhere | Considers the parameters like n/w bandwidth, storage capacity *etc.* that make it less reliable | Random walk approach make it reliable in case the resources are equally distributed |
| *Complexity* | $\mathcal{O}(logn)$ | $(\log\log n)$ | $\Theta(n)\frac{1}{2}$ | $\Omega(n)$ | $\Omega(log\_2n)$ | $\Theta(n)$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| *Manageability* | Complex Architecture hence difficult to manage | Quite easy to manage as lots of its working is dependent on single node | Management is easy due to SDRT algorithm for dealing with the different topologies | Manage the consistency by using the data dissemenination algorithm | Uses Algorithm DIAR for Resource Discovery | Better management can be achieved by combining its two approaches |
| *Adaptability* | Multiple platforms environment ake it more adaptive | Can be made adaptable by providing managerial information about different platforms | Routing table is used to make records of different platform hence, adaptability increases | Adaptable due to universal, network and distractive awareness parameters | Depends upon the Service Level Agrement(SLA) sign with user for providing Adaptability | By Using the best neighbour Approach adaptability is easy |
| *Development Agent* | DHT | OWL, RDF | *C* | *C*, Java | Any Description Language | *C* |
| *Algorithm Used* | Swaming | Matchmaking, Gang-match-making | SDRT, Routing | Dissemination | DIAR | Request Forwarding Algorithm |
| *Client Server* | Each node is Client as well as server | Yes | Yes | Yes, as ut is location independent | Yes | Yes |

| | | | | | | |
|---|---|---|---|---|---|---|
| *Extensibility* | Easily Extendable | Easily | Complex | Easily Extended due to the distinctive awareness | Easily as well as complex | Easily Extendable |

Shy Kutten and David Peleg, proposed an algorithm named "Deterministic Algorithm" [SK00]. They have proved that their algorithm was optimal in all the measures that are time, message and communication complexities. A Resource Discovery based on peer-to-peer model has been proposed, which consists of a few request-forwarding algorithms in a fully decentralized architecture accommodating heterogeneity and dynamism in resources [IF01b]. Following algorithms are described in detail as given below; Flooding Algorithm [MHBL99], Swamping Algorithm [MHBL99], Random Pointer Jump Algorithm [MHBL99], Name Dropper Algorithm [MHBL99], Absorption (Randomized Resource Discovery) Algorithm [CL00], Data Flooding Based Data Dissemination Algorithm [MHBL99], and Discovering Intermittently Available Resources (DIAR) Algorithm [YH02].

### Flooding Algorithm

The flooding algorithm is used by routers in the Internet, broadcasting by local queries, known as "gossiping" [SHL88], [Pel96] have been used to maintain consistency in replicated databases [DAS97] and to gather information about system failures [AK83]. This algorithm is based on agent-based approach [MHBL99]. It is also used in peer to peer file sharing system, advertising certain routing tables or link states to routers and some part of routing protocols that are used in ad-hoc wireless networks. Every node acts as a transmitter and receiver and every node tries to send every message to every node of its neighbor. Newly added edge is not used for any communication. Direct communication exists only in between initially

existing set of neighboring edges of the network. The required number of rounds of this algorithm is equivalent to the diameter of the graph. This algorithm can be very slow if not started with a graph, which has small diameter. This algorithm is not quite good for Grid Computing, due to the size of the Grid environment which, consists of several nodes. This algorithm may increase the traffic congestion in the network and the overall performance of the Grid system may be slow.

### The Swamping Algorithm

Agent based approach acts as the base approach for implementing this algorithm. Swamping algorithm is identical to flooding algorithm except that this algorithm allows a node to connect not only with the set of initial neighbors but also with all of its current neighbors [MHBL99]. The main advantage of this algorithm is that it has better speed than the flooding algorithm and it needs $\mathcal{O}(logn)$ rounds to converge to a complete graph. The disadvantage of this algorithm is, its communication complexity grows very quickly. In summarization of this algorithm, it is very fast to complete the graph using less rounds, however the communication complexity is more terrible: as it grows very quickly. This algorithm improves the flooding algorithm by reducing the number of rounds. Hence, its traffic congestion on the network is higher than the flooding algorithm.

### The Random Pointer Jump Algorithm

In this algorithm, in each round, each node contacts with a random neighbor, and then this random neighbor sends all of its neighbors to the sender node. Finally sender neighbor and random neighbor's neighbors get merged [MHBL99]. This algorithm claimed that a strongly connected graph with $n$ nodes needs $\mathcal{O}(n)$ complexity time to converge to a complete graph as shown in Figure 2.5 and its good choice for strongly connected graphs

There are two kinds of random pointer jump algorithms.

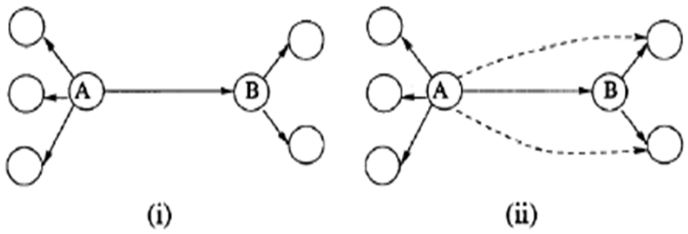



Figure 2.5: Random Pointer Jump

(i) Before the Random Pointer Jump: as shown in Figure 2.5(i), Node A chooses at random one of its neighbors and opens a connection with it. Here B is the chosen neighbor.
(ii) After the Random Pointer Jump: Figure 2.5(ii) shown that, Node B has passed to node A all of its neighbors, and now A also points to them.
The dashed lines indicate newly formed edges. A more advanced version of this algorithm is "Random Pointer Jump with Back Edge". This algorithm is almost identical to Random Pointer Jump algorithm except every time adding a back edge after performing the pointer jump. Both of the above versions of random pointers algorithm uses the agents to control the query processes and to make the Resource Discovery decisions based on their own internal logics.

### Name Dropper Algorithm

Mor Harchol et al. proposed a new algorithm for querying resources in a weakly connected network where it is assumed that all machines already know each other and named it "Name-Dropper" [MHBL99]. The algorithm, which takes $\mathcal{O}(log^2 n)$ rounds to learn each other, $\mathcal{O}(n^2 log^2 n)$ number of connections and $\mathcal{O}(n^2 log^2 n)$ number of pointers in a network. In [MHBL99], the authors claimed that this new algorithm

is more efficient in terms of required time and total number of network communications compare to four other algorithms, mainly flooding algorithm, swamping algorithm, random pointer jump algorithm, and random pointer jump with back edge algorithm [MHBL99] and also claimed that this algorithm achieves "near-optimal performance both with respect to time complexity and with respect to the network communication complexity". This algorithm is almost identical to "Random pointer" algorithm. But according to [MHBL99], in worst-case $\mathcal{O}(log^2 n)$ rounds of "Name-Dropper" algorithm dominates over $\Omega(n)$ rounds of complexity time of "Random pointer" algorithm, where $n$ is the number of machines.

**A Time-to-Live based reservation algorithm**

Kenji and Toshitsgure gave the new algorithm named "A time-to-live based reservation algorithm on fully decentralized Resource Discovery in Grid Computing" [STY05]. This algorithm is based on a simply unicast request transmission that can be easily implemented. The addition of a reservation algorithm enables Resource Discovery mechanism to find more available matching resources. The deadline for Resource Discovery time is decided with time-to-live value. The performance of algorithm is compared with Iamnitchi [Iam02] and author claimed that it has the better performance based upon the first-found-first-served technique.

**Deterministic Resource Discovery Algorithm**

S. Kutten and Peleg proposed a better algorithm named "deterministic Resource Discovery" [SK00]. By comparing the algorithm with name dropper and randomized algorithm time complexity is reduced from $\mathcal{O}(log^2 n)$ to $\mathcal{O}(logn)$, for message complexity this improved algorithm takes $\mathcal{O}(nlogn)$ compare to $\mathcal{O}(log^2 n)$ and this algorithm also takes $\mathcal{O}(|E_0|log^2 n)$ complexity for communication compare to $\mathcal{O}(n^2 log^3 n)$.

### Absorption (Randomized Resource Discovery) Algorithm

C. Law et al. proposed a randomized Resource Discovery algorithm called "absorption algorithm" based on a strongly connected graph and this algorithm is developed in two stages [CL00]. In stage 1, a network of $n$ nodes is partitioned into clusters; each cluster has its leader node and all nodes in a cluster know their leader. In this stage, an ultimate leader is determined in the network. In stage 2, the ultimate leader of the network broadcasts the pointers to the each member of the network in one time step. The authors claimed that this absorption algorithm takes $\mathcal{O}(logn)$ steps time complexity, can send $\mathcal{O}(nlogn)$ number of messages, and can pass $\mathcal{O}(n^2logn)$ number of pointers with high probability.

### Distributed Awareness Algorithm

After analyzing different algorithms MorHarchol et al. in [MHBL99], proposed a new agent-based Resource Discovery algorithm called "Distributed Awareness Algorithm" and they also proposed a framework for dynamic assembly of agents based on this algorithm. Distributed awareness refers to a learning mechanism by which a node gets awareness about other nodes in a network. In this algorithm, each node has an awareness table and each node exchanges the information of this awareness table with other nodes. A typical awareness table entry contains location of the node (IP address); when last heard from the node, when last time the awareness information was sent to the node. The authors claimed that this agent based Resource Discovery system could provide better discovery services using its agents autonomous behavior.

### Data Flooding Based Data Dissemination Algorithm

A algorithm named "Data Dissemination Algorithm" which follows swamping approach for message distribution is given by Maheswaran et al. [MK00]. When a message comes to a node, that message gets validated. This validation process relies

on three types of disseminations as discussed earlier in parameter based approach, *universal awareness* that permits all incoming messages, *neighborhood awareness* that allows messages from a certain distance, and *distinctive awareness*, which discards messages which has less Grid potentiality at the local node. Maheswaran, had also measured the performance of "universal awareness", "neighborhood awareness", and "distinctive awareness" dissemination schemes. The authors claimed that universal approach is more expensive in terms of message complexity than that of neighborhood and distinctive approach. The authors claimed that this new class of dissemination could reduce the communication overhead during the Resource Discovery process.

**Discovering Intermittently Available Resources (DIAR) Algorithm**

This algorithm is used to discover the occasionally available resources in multimedia environment [YH02]. Different policies for a QoS based Resource Discovery service for a given graph theoretic approach is defined. A generalized version of Discovering Intermittently Available Resources (DIAR) algorithm based on occasionally available resources has been discussed in [YH02].

In this section we have reviewed the models, approaches and existing algorithms for Resource Discovery in large-scale heterogeneous environment. Based on the review work the next section will addresses the Resource Discovery in existing middleware. Resource Discovery, in the Grid environment, becomes complex as the resources are geographically distributed, heterogeneous in nature and owned by different individuals and organizations each having their own resource management policies and different access and cost models [RBD00].

## 2.3 Resource Discovery in Existing Middleware

Recently middleware are reemerged as a means of integrity software applications running in distributed heterogeneous environment. Formally, Grid Middleware can be defined as: *"A mediator layer that provide a consistent and homogeneous access to resources managed locally with different syntax and access methods"* [ML05]. Till today, various implementations of the Grid Middleware exist in the market. But most notable is the *"Globus"* that is explained below, which has defined the de-facto standard for Grid Middleware [Tea01]. Other efforts are SUN N1GE6, Condor and Alchemi. Each Middleware has different aspects on architecture and designing. Thus, it creates a lot of confusion for the users to select a Middleware or any component of the Middleware for creating the Grid environment.

### 2.3.1 Globus Toolkit

The Globus Toolkit is collection of tools that provides the basic services and capabilities like Resource Management, Resource Discovery, Security, Information Services *etc.* that are required for Grid Computing. It is designed to enable people to create computational Grids. Globus is most widely utilized and explored infrastructure software for Grid middleware development, worldwide among Grid practitioners [JF04]. It has been developed over several years chiefly at the Argonne National Laboratory Illinois USA. Globus toolkit is an open source set of services and software libraries that supports the development of Grid systems and applications [FK97]. As an open source project, any person can download the software, examine it, install it and improve it. Globus architecture has provided the basic building blocks to do most of the things like software for security, information infrastructure, resource management, data management, communication, fault tolerance and portability which can be acknowledged as globus pyramids.

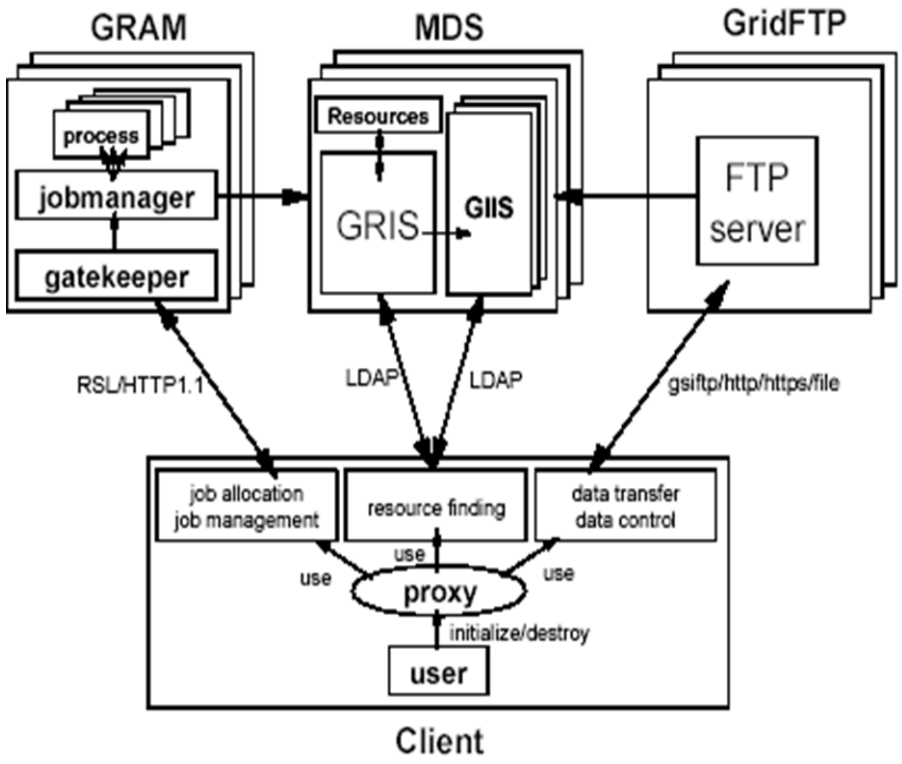



FIGURE 2.6: System Overview of Globus Toolkit

### Components of Globus Toolkit

For each pyramid mentioned in above section, Globus provides a component to implement resource management, data management, and information services, as illustrated in Figure 2.6.

- GRAM/GASS
- GridFTP
- GSI
- MDS(GRIS/GIIS)

- **GRAM/GASS**

  The primary components for the resource management pyramid are the Grid Resource Allocation Manager (GRAM) and the Global Access to Secondary Storage (GASS).

- **GridFTP**

  GridFTP is a key component for the secure and high-performance data transfer. The Globus Replica Catalog (GRC) and Management is used to register and manage both complete and partial copies of data sets. These three pyramids are modularized and can function in isolation; however, together, they complementary to each other.

- **GSI**

  All of the above components are built on top of the underlying Grid Security Infrastructure (GSI). This provides security functions including single/mutual authentication, confidential communication, authorization, and delegation.

- **MDS(GRIS/GIIS)**

  Based on the Lightweight Directory Access Protocol (LDAP), the Grid Resource Information Service (GRIS) and Grid Index Information Service (GIIS) components can be configured in a hierarchy to collect the information and distribute it. These two services are called the Monitoring and Discovery Service (MDS). In particular, Resource Discovery is addressed by the MDS, which provides a framework for publishing and accessing information about the Grid resources [ML05]. MDS defines and implements the mechanism for service and Resource Discovery and monitoring in distributed environments [ALV05].

Monitoring process is used for observing the Grid resources, fixing problems and tracking resource usage. Discovery process is used for finding suitable Grid resources information to perform the task. The information collected can be static information about the machines as well as dynamic information showing current CPU or disk activity. A rich set of information providers is included with the Toolkit and the Globus users can add their own. The information provides an interface with the GRIS, which reports this information to a hierarchy of GIIS servers in the Grid. The LDAP query language is used to retrieve the desired information. Basically, MDS contains the following components:

- Grid Resource Information Service (GRIS)
- Grid Index Information Service (GIIS)
- Information Provider
- MDS client

In Globus Toolkit2 (GT2), the MDS2 provides information about the system component using the LDAP as a uniform interface for such information. In GT3, MDS3 was implemented. Like MDS2 it has also the hierarchical architecture, but it has been redesigned to be compliant with Open Grid Service Architecture (OGSA) model [IFT03]. In GT4, MDS4 was implemented. MDS4 provides a web service resource framework [KCV04]. Figure 2.7 shows the architecture of the Globus MDS2.

### 2.3.2 Alchemi

Alchemi is .NET-based enterprise Grid computing and runtime machinery for creating a high-throughput resource-sharing environment [ALV05]. The Alchemi Grid

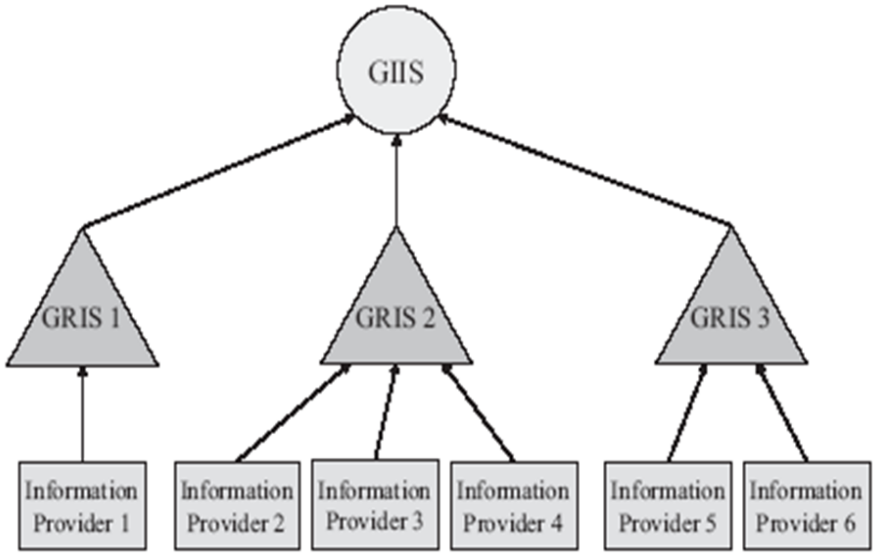



Figure 2.7: Architecture of Globus MDS2

Computing framework was conceived with the aim of making Grid construction and development of Grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility [SV04]. Alchemi architecture follows the master-worker parallel programming paradigm [KW84] in which a central component dispatches independent units of parallel execution to workers and manages them. Alchemi includes:

- The runtime machinery (Windows executables) to construct computational Grid.
- A .NET API and tools to develop .NET Grid applications and Grid-enabled legacy applications.

### Alchemi Components

Various key components of Alchemi architecture as shown in Figure 2.8 are:

- Manager

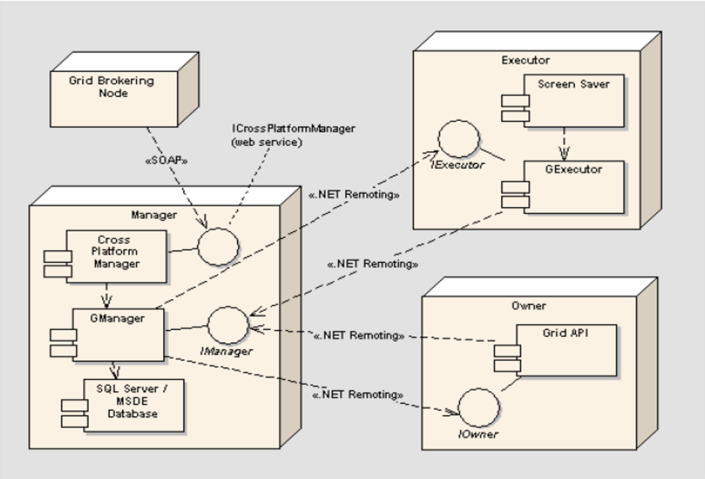



FIGURE 2.8: Alchemi architecture and interaction between its components [SV04]

- Executor
- Owner
- Cross-Platform manager

- **Manager**

  It logically couples the Windows Desktop machines running the instance of Alchemi Executor service. The Manager manages the execution of Grid applications and provides services associated with managing thread execution.

- **Executors**

  The executors register themselves with the Manager, which in turn keeps track of their availability. Threads received from the owner are placed in a pool and scheduled to be executed on the various available executors. Grid applications created using the Alchemi API are executed on the owner component.

- **Owner**

  It provides an interface with respect to Grid applications between the application developer and the Grid.

- **Cross-Platform Manager**

  Cross Platform Manager is an optional sub-component of the Manager. It is a generic web service interface that exposes a portion of the functionality of the Manager to manage the execution of platform independent Grid jobs.

Similar to Globus, Alchemi is an open source software framework that allows us to painlessly aggregate the computing power of networked machines into a virtual supercomputer (Computational Grid) and to develop applications to run on the Grid. As Alchemi is the emerging technology, so a lot of work has to be done to make it a standard .NET based middleware. In context to Resource Discovery part in the alchemi, executor can only be working on one thread at a time, so the manager sends threads to the idle executors, based upon the order each one becomes idle and not based upon the load of the machine running the executor. The executors are manually configured to connect to the Managers. As to actually modifying the selection of which executors to use for a task, and, say, abandoning tasks sent to 'slow' executors, that's something that hasn't been touched yet.
The manager, using the First Come First Serve (FCFS) Algorithm, whenever the application to be executed it needs resources, it requests the manager and the manager assigns the resource what ever is there on basis of FCFS. The key features supported by Alchemi are:

- Internet-based clustering [NNN98] of desktop computers without a shared file system [YAT99].

- Federation of clusters to create hierarchical, cooperative Grids.

- Dedicated or non-dedicated (voluntary) execution by clusters and individual nodes.
- Object-oriented Grid thread programming model (fine-grained abstraction).
- Web services interface supporting a Grid job model (coarse-grained abstraction) for cross-platform.
- Interoperability example for creating a global and cross-platform Grid environment via a custom.
- Resource broker component.

### 2.3.3 Condor

Condor is resource management systems for compute intensive jobs, which maintains the High Throughput Computing (HTC) resource [IF01a]. The main function of Condor is to harness the idle CPU cycles on heterogeneous pool of computers [RR98]. Condor is a specialized workload management and high-throughput distributed batch computing system. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, Resource Discovery, and resource management [Jul04]. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion [DTL05].

#### Condor Components

Various key components of Condor architecture as shown in Figure 2.9 are:

- Resource-owner Agent
- Customer Agent

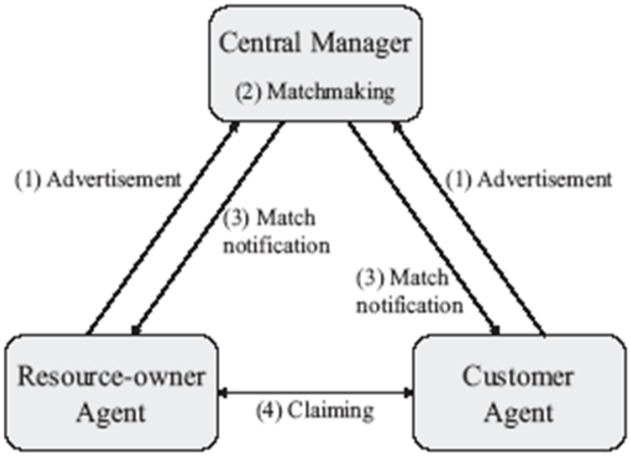



Figure 2.9: Architecture of Condor

- Central Manager
- **Resource-owner Agent**

  Resource owner agent, submits resource requests (job requirements) in an application queue ordered by a priority scheme. Any machine in the pool (including the Central Manager of the pool) can be configured for whether or not it should allow Condor jobs to be submitted. The resource requirements for a submitter machine are actually much greater than the resource requirements for an executer machine. Therefore, if a job have a large memory image and user wants to submit them, then a large amount of disk space is needed to hold these jobs.

- **Customer Agent**

  Customer Agent periodically extracts resource state information and updates its resource offers. Any machine in pool (including Central Manager) can be configured for whether or not it should execute Condor jobs. Obviously, some of the machines will have to serve this function or machine pool won't be very useful. Being an executer machine doesn't require many resources at all. About the only resource that might matter is disk space, since if the remote job

dumps core, that file is first dumped to the local disk of the execute machine before being sent back to the submit machine for the owner of the job.

- **Central Manager**

  This is the kernel of the Condor Pool. There can be only one central manager for condor pool. The machine is the collector of information, and the negotiator between resources and resource requests. Central Manager plays very important role in the Condor pool. The central manager will ideally have a good network connection to all the machines in the pool, since they all send updates over the network to the central manager. All queries go to the central manager.

  Condor adopts a centralized scheduling model. Central manager performs the scheduling task. Condor is able to effectively harness the power of non-dedicated resources by using unique features such as check pointing, job migration, remote system calls, and ClassAds specification language. ClassAds are used extensively in the Condor system to represent jobs, resources, submitters and other Condor daemons.
  ClassAds are a flexible mechanism for representing the characteristics and constraints of machines and jobs in the Condor system.

### 2.3.4 Sun N1GE6

Sun Grid Engine (SGE) [Eng90] is batch-queuing system and Resource Management software for clusters and Grids. Sun Grid Engine is new generation distributed Resource Management system which dynamically matches users hardware and software requirements to the available (heterogeneous) resources in the network, according to policies usually defined by management in an enterprise [Gen01].

To quote SGE's open source website, *"the Grid Engine project is an open source community effort to facilitate the adoption of distributed computing solutions"*. Sponsored by Sun Microsystems and hosted by CollabNet, the Grid Engine project provides enabling distributed resource management software for wide ranging requirements from compute farms to Grid Computing. The basic motivation behind SGE is to provide precise control over resource usage and maximize utilization in clusters, where the resources are idling most of the time otherwise [ea06a]. Architecture of Sun N1GE6 is depicted in Figure 2.10.

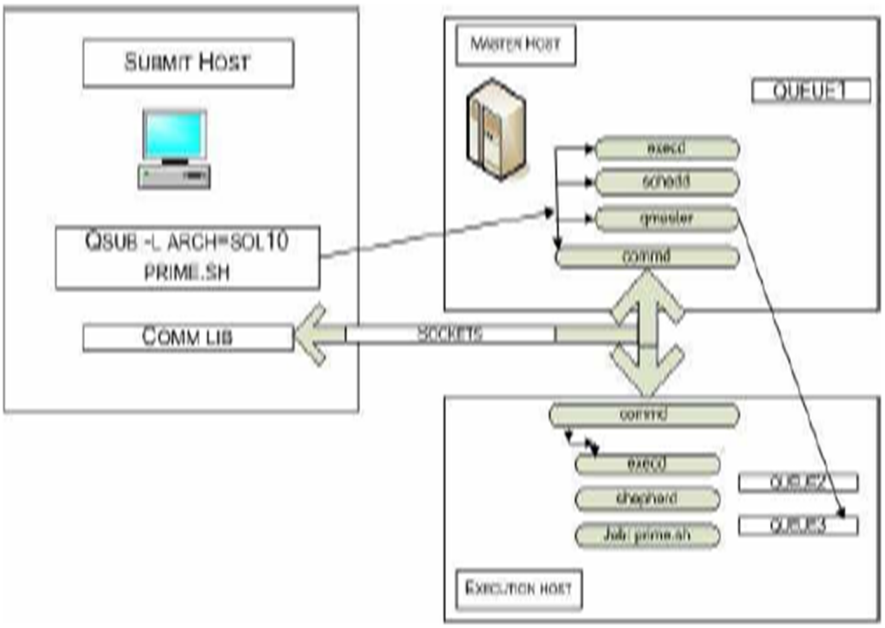



Figure 2.10: Architecture of SUN N1GE6

### Sun N1GE6 Components

Various key components of SUN N1 GE6 architecture are:

- Hosts
- Daemons

- Queues
- Complexes
- Job Submission

N1GE6 consist of the collection of Hosts and Daemons:

- **Hosts**

  A computer system that is accessed by a user or contains the data. The *master host* acts as the core of this centralized system. An *execution host* is a node within the cluster that is capable of executing jobs. *Administration hosts* are nodes that are given permission to alter the configuration of the Grid Engine cluster. Also described in [HS95] *Submit hosts* are nodes in which users are allowed to submit jobs from to the Grid Engine. *Shadow hosts* monitor the master host. If the master host fails, the shadow host takes over.

- **Daemons**

  These are processes that run on nodes of the Grid Engine cluster [Eng90]. *Master Daemon* (called qmaster daemon) is central to the Grid Engine. *Scheduler Daemon* (schedd) is responsible for retrieving the current state of the cluster from qmaster and deciding which queue a job should be assigned to. *Execution Daemon* (execd is) responsible for running jobs for the queue(s). In Shepherd Daemon an individual shepherd is started by the execd as the parent process of each job. *Communication Daemon* (commd) is used to separate communication from processing. *Shadow Daemon* (shadowd) monitors the qmaster daemon. If the shadowd detects the qmaster has failed all running shadowd decide which is to take over as the new qmaster.

- **Queues**

  A queue in N1GE6 represents a class of jobs allowed to execute concurrently on an execution host [Eng90]. Jobs do not wait in a queue, once dispatched to a queue the job is associated with that queue and runs on that execution host.

- **Complexes**

  The complex configuration provides all pertinent information about the resource attributes users can request for jobs with the qsub -l or qalter -l commands. The complex configuration also provides information about how the Grid engine system should interpret these resource attributes [Mic05].

- **Job Submission**

  Users of Grid Engine can submit its jobs from any of the submit hosts of the Grid Engine cluster. The command line tool qsub can be used or a job can be submitted using the GUI qmon. Job submission process of Sun N1GE6 is shown in Figure 2.11. The SGE is a new generation distributed resource management and scheduling system from Sun Microsystems that can be used to optimize the utilization of software and hardware resources in a UNIX-based computing environment. The SGE can be used to find a pool of idle resources and harnesses these resources; also it can be used for normal activities, such as managing and scheduling jobs onto the available resources. These all processes are maintained by JXTA.

JXTA [BTY02] [Wat01] [JXT02] technology is a set of open, generalized peer-to-peer protocols that allows any connected device (cell phone to PDA, PC to server)

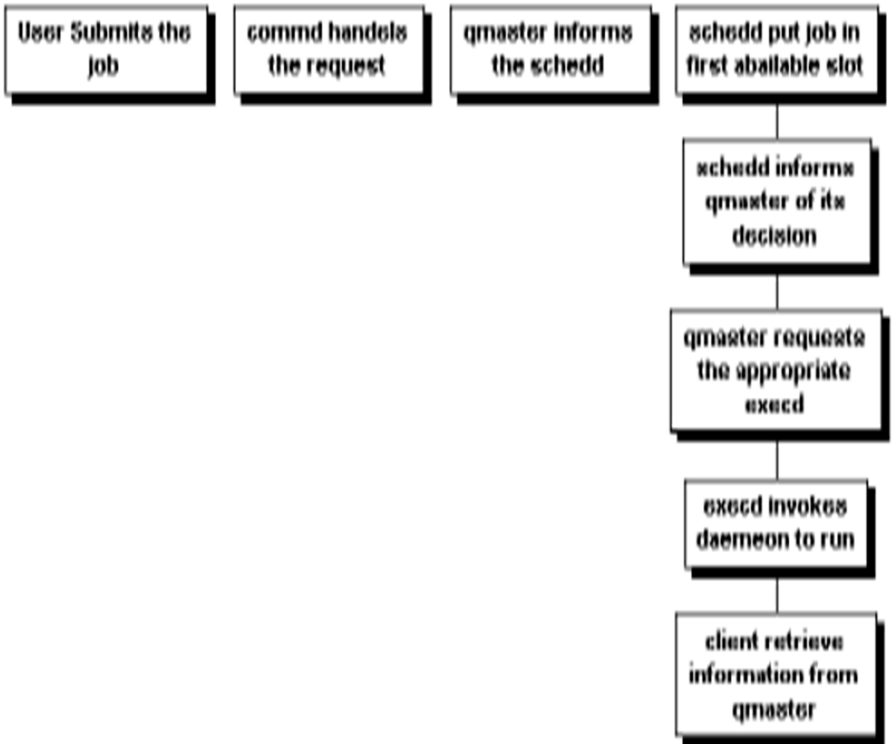



FIGURE 2.11: Job submission process in SUN N1GE6

on the network to communicate and collaborate. JXTA technology enables developers to create innovative distributed services and applications. JXTA is an open source effort that involved the developer community from the start. JXTA defines six stateless best-effort protocols for ad hoc, pervasive, and multi-hop P2P computing [Hos02].

JXTA has layered architecture. Different layers that are used for different functionalities are:

*JXTA Core* It includes the Peer groups, Peer IDs & security peer monitoring.

*JXTA Servies:* It includes the discovery services, indexing services and membership services.

*JXTA Applications:* It includes file sharing & resource sharing.

## 2.4 Comparison of Resource Discovery in Existing Middleware

Section 2.3 discussed Resource Discovery in existing Middleware. A comprehensive Comparison of all Middleware including; Architecture, scalability, reliability, adaptability, security and manageability has been done and placed in Table 2.2.

## 2.5 Problem Formulation

Resource Discovery in large distributed network environments, like in a Grid environment, needs to be very efficient and optimized for a number of parameters. Efficient and fast discovery is very critical and challenging since the underlying mechanism needs to take care of heterogeneous resources, varying resource requirements of the large number of users and account for dynamic status of resource availability. For making Resource Discovery in Grid environment extremely efficient; practical and optimized discovery algorithms, specifically designed to take care of these factors are needed.

As observed during literature review in this chapter various models, approaches and algorithms exist for discovering the resources. The different approaches include query based, agent based, peer-to-peer, ontology description based, QoS based and parameter based approaches. Different algorithms based on these approaches have also been mentioned including flooding algorithm, swamping algorithm, random pointer algorithm, DIAR algorithm *etc.* However, these algorithms have not been proved to be efficient enough to take care of heterogeneity of resources; dynamic availability of resources, user's specific needs and other such desired factors.

Thus, there is need a for identification and analysis of the specific requirements,

dynamic decision factors and desired QoS parameters that need to be addressed during Resource Discovery process. Further, based on this analysis, the focus will be to design and implement resource discovery algorithms that work effectively in generic and specific Grid environments.

Keeping this in view, firstly the Resource Discovery problem has been reformulated as a graph theoretic problem. Reformulating the problem in this way gives a robust base to solve it in optimized manner based on given constraint and parameters. Another advantage will be that existing graph theoretic solutions can also be explored to solve the reformulated resource discovery problem.

In second, *i.e.* ontology description based approach; a service matchmaking mechanism based on ontology knowledge is used. The matchmaking framework will be evolved to provide a optimized Resource Discovery and also to provide close matches. The effort in both the approaches will be to identify and optimize QoS parameters for Resource Discovery in Grid environment, that addresses efficient Resource Discovery requirements related to Time, Cost, Reliability and Security.

## 2.6 Objectives of the thesis

Followings are the objectives of the thesis:

- To identify and analyze the specific needs, dynamic decision factors and desired QoS parameters that need to be addressed during Resource Discovery process in Grid environments.
- To design and implement Resource Discovery algorithms that work efficiently in specific and/or generic Grid environments.
- To test and validate the proposed algorithms in simulated and actual Grid

environments.

- To demonstrate the usability of proposed Grid Resource Discovery algorithms for a set of constraints, Quality of Service (QoS) parameters and applications.

The objective stated above have been accomplished as per following methodology:
*Objective (1) To identify and analyze the specific needs, dynamic decision factors and desired QoS parameters that need to be addressed during Resource Discovery process in Grid environments.*

To accomplish the first objective, a comprehensive study of different QoS parameters (like time, cost, reliability, security *etc.*), dynamic decision factors and specific needs of QoS in Resource Discovery process have been made. Detailed analysis of different QoS parameters has been done. A thorough study of different approaches and algorithms used in QoS aware Resource Discovery is also carried out. Work done in the area of Resource Discovery process has been studied in detail.

*Objective (2) To design and implement Resource Discovery algorithms that work efficiently in specific and/or generic Grid environments.*

During literature survey it became apparent that different types of models, algorithms and approaches are required for the Resource Discovery process. Three different algorithms for efficient Resource Discovery have been proposed, designed and implemented. These algorithms are easily parallelized and efficient in terms of time, cost and matchmaking principles.

*Objective (3) To test and validate the proposed algorithms in simulated and actual Grid environments.*

To achieve the third objective, we have used Microsoft Visual Studio 2005 as

front-end and MySQLServer as back-end to show the performance of algorithms and observe QoS parameters.

*Objective (4) To demonstrate the usability of proposed Grid Resource Discovery algorithms for a set of constraints, Quality of Service (QoS) parameters and applications.*

To achieve the fourth objective, firstly: A Grid testbed namely *"TUGrid Tesbed"* has been designed and created. Secondly, all the three algorithms have been deployed on TUGrid testbed. The algorithms have been tested and the experimental results obtained are reported, along with the complete testbed layout, configuration details and results obtained.

## 2.7 Conclusions

This chapter highlighted the Resource Discovery process followed by in depth of its models, approaches and algorithms. A comprehensive study of different middleware also have been done. Finally, gap areas have been identified and based upon it problem is formulated. After the problem formulation, the set of objectives have been laid down. Next chapter, *Identification and Analysis of Quality of Service Parameters,* focuses on the QoS aware Resource Discovery and highlights the case study of the campus Grid.

Table 2.2: Comparison of Related Middleware

| *Middleware/Property* | *NAGE6* | *Condor* | *Alchemi* |
|---|---|---|---|
| *Developer* | High scalability as it uses the four axes framework | Less Scalable as Broker is centeralized | Scalable as it uses the routing protocol |
| *Architecture* | Hierarchial | Hierarchial | Centralized |
| *Resources* | Unix like OS, Windows (execution node only) | Extensible schema model, hybrid namespace, no QoS, n/w directory store, centeralized query discovery, periodic push dissemination | Windows |
| *Platform* | Unix like OS, Windows (execution node only) | Linux, Windows | Windows |
| *Reliability* | More reliale due the presencde of shadow master host as a source of bbackup | Higly reliable for single purpose and single application envionment | Flexible application composition by supporting OO application programming and file based Job model |
| *Security* | Highly Secure PKI , X 509, Authentication as well as Authorization | Defined authentication, data integrity and authorization models | Security is due to powerful toolset of .NET framework |
| *Threaded Programming* | No, | No | Yes |
| *Sheduling* | Uses kernel and user level Checkpointing, | Centeralized Schedular | Uses FCFS |
| *Application Type* | HTC, Computational, | Computational | HTC |

| | | | |
|---|---|---|---|
| *Implementation Technologies* | DRMAA-C,JAVA,PERL Bindings | *C*, JAVA | ♯, Web Services, .NET framework |
| *Inteoperability* | Interoperate between PBS, Condor, LSF. Using tool given by Grid Wise Tech | Interoperability with Globus due to the support for MPI and PVM | Using Grid Bus Broker Integrate with any Middleware |
| *Fault Tolerance* | Uses kernel and user level Checkpointing, | Uses system level Checkpointing | Uses Heartbeating mechanisms for dynamically updating its status of executors |
| *Ease of Use* | Provides GUI using QMON ans SUN Grid Portal | Not Easy as it is based on CUI | Easy to use As GUI and Widows based platform |
| *Resource Discovery* | Uses the JXTA and DRMAA components | Uses the matchmaking Mechanism | Based upon the FCFS |
| *Performance* | The JXTA support makes it fast discovering the resources | Performance is high as depend on the Class-Ad mechanism | Is limited for small loads due to the absence of scheduling, discovery and fault tolerance mechanisms |

# 3

# Identification and Analysis of Quality of Service Parameters

*In previous chapter, "Literature Review", detailed study of Resource Discovery process, its components and characteristics are explained. Further, Resource Discovery models, approaches, and algorithms have been explained. Finally a comprehensive review of the work done in Resource Discovery is reported. Based upon the literature review, gap areas have been identified where further improvements are needed. This gap analysis gives us a basis for problem formulation in the same chapter. During the problem formulation, the set of objectives have been framed. This chapter is based on the first objective, i.e. "To identify and analyze the specific needs, dynamic*

*decision factors and desired Quality of Service (QoS) parameters that need to be addressed during Resource Discovery process in Grid environments". Organization of the chapter is as follows: Section 3.1 focuses upon the QoS in Grid Computing. Section 3.2 describes the Identification of QoS parameters followed by; QoS Specification, Service Level Agreement (SLA), SLA lifecycle and QoS classification for Grid-aware reconfigurable applications in QoS. Section 3.3 deals with the Analysis of QoS Parameters in Resource Discovery process. Then a detailed description of the QoS aware Resource Discovery using SLA is explained by taking the case study of an organization in Section 3.4. Proposed design and Implementation description have been presented in Section 3.5*

## 3.1 QoS in Grid Computing

With numerous emerging real-time components in Grid Computing applications, there has been interest for developing mechanisms that enable real-time services over the Grid to a large extent. Grid applications require more sophisticated management of those system components, which affect the Quality of Service (QoS) delivered to the users, than for data-only systems. QoS is a concept in which clients can indicate the level of service they require. For example in real-time voice communications, the clients prefer reliable delivery times over guaranteed delivery and in financial applications, a client may prefer encrypted communication over the faster communication.

QoS mechanism in Grid Computing requires to support more than guaranteed availability of given resources. QoS guarantees in such a context not only the behavior of the single components of the Grid system, *i.e.* processor availability on a cluster computer, but also defines, how the whole Grid environment manages the

given job [LOB05]. According to Edgar and Joan Different levels of QoS like "diamond" (efficiency guaranteed), "gold" (completely distributed) "silver" (searching alternatives) "bronze" (best effort) *etc.* are explained in [EM05].

From the end-user's point of view, QoS should be supported end-to-end between any pair of hosts; so all elements along the path must participate in special treatment for packets to provide the required QoS. In Grid Computing QoS is not limited to the network bandwidth but also extends to the processing and storage capability of the nodes. QoS is directly related to performance of the Grid that a given client experiences when invoking on a remote server [DC06]. The Grid QoS system should provide strategy and functions that can convert and map the QoS parameters of logical resources to that of physical resources [DAM04].

Different QoS parameters are; Time, Accuracy and Integrity are discussed by Cris et al. in [CS97]. Where as according to Rimantas et al. some of the critical parameters are Availability, Accessibility, Latency, Latency Fluctuation *etc.* QoS can be viewed as providing assurance on a set of qualitative characteristics as discussed above such as Time, that is needed to complete the individual task and deviation from ideal rate of event occurrence; Accuracy, is based on the difference between the actual and expected service values [SBM04]. Integrity defines the different constraints to use the resources in terms of QoS. Cost of a single resource and multiple resources; Security, for increasing the performance; Reliability, that is necessary to execute the grid applications; Availability is a parameter showing if the system is ready to provide the resource immediately. This is the probability that resource is available at a particular moment; Accessability is a percent of time, when the system is functional for an end user. The system may be available but not accessible; Latency is a difference between the moment when the first request is passed an input and the moment when the last request passed the output checkpoint at the otherend;

Latency valuers depends on the throughput of the intermediate internet. Latency fluctuation is the time span between minimum and maximum latency values at the same cluster.

## 3.2 Identification of QoS Parameters

Today's computing environments are becoming more and more distributed in nature. At the same time, the applications used in these environments are becoming more complicated and are being used in more mission critical roles in the enterprise. Consequently, user's demands for performance, reliability, accuracy, integrity, usability and availability are increasing rapidly. To meet these needs, a high level of Quality of Service (QoS) must be delivered to the user. Doing so, however, is not an easy task [MJKB00]. To continue work in this area, there must be an access to a tightly controllable, highly portable, and flexible environment suitable for QoS experimentation. Such an experimental testbed is not only useful for gaining valuable insight into developing better QoS solutions, but it is also required for evaluating the merit of these solutions. Different mechanisms (schemes) those are necessary to identify these parameters are QoS specifications, SLA, QoS classifications *etc.* as described below:

### 3.2.1 QoS Specifications

QoS specifications for efficient Resource Discovery are the high-level description of expectations given either by the user or obtained from application requirements [KK06]. Following are the factors which, user's can specify for QoS requirements:

- Performance - The expected performance characteristics are needed to establish resource commitments.

- Synchronization - It characterizes the degree of synchronization required between related services, events, or information flows.

- QoS Management - The degree of QoS adaptation that can be tolerated and scaling actions to be taken in the event when the contracted QoS cannot be met.

- Cost of service - It is the price of a resource when a user is willing to obtain a level of services.

- Level of Service - It specifies the degree of resource commitment required to maintain performance guarantees.

QoS specification is followed by the notion of Service Level Agreement (SLA) [Dob04] and Resource Level Agreements (RLA) to guarantee QoS. Each service establishes a RLA with the resources they use and executed by a set of resources. SLAs provide guarantees at the service level and detail description of same is described below.

### 3.2.2 Service Level Agrement (SLA)

SLA is an agreement between the service provider and a user that specifies in measurable terms the services provided by the service provider. Also, when multiple service providers are present, SLA controls resource allocation among these providers for the different services. This agreement implies that the user will be able to discover new service provider during failure of the existing service providers and recovery of services by new alternate service composition when the resource(s) fails. The service provider reads the user/application requirement specifications and translates/maps them into QoS requirements [KK06].

### SLA Lifecycle

SLA quantitatively defines the performance level of service requested and the obligations for the parties involved. It includes:

(*i*) Identities involved in contract.

(*ii*) Penalties applicable to the parties.

SLA provides a simple abstraction of resources on the Grid. SLA Life cycle consist of five phases:

- SLA Template Development: In this phase the SLA templates are developed.
- Negotiation and sales: In this phase the SLA is negotiated and the contracts are executed.
- Implementation: During this phase the SLA is generated.
- Execution: The SLA is executed, monitored, and maintained.
- Assessment: Evaluation of the SLA performance. In this phase, a reevaluation of the initial SLA template might be done.

### Types of SLA

When identification of different QoS parameters has to be done it is very mandatory to check it out which kind of agreement is going to signed on. There are basically three types of SLAs are:

- Task SLA (TSLA): This is the performance of the activity or the task. A TSLA is created by submitting a job description to a queuing system.
- Resource SLA (RSLA): This is the right to consume a resource without specifying what it will be used for example advance reservation.

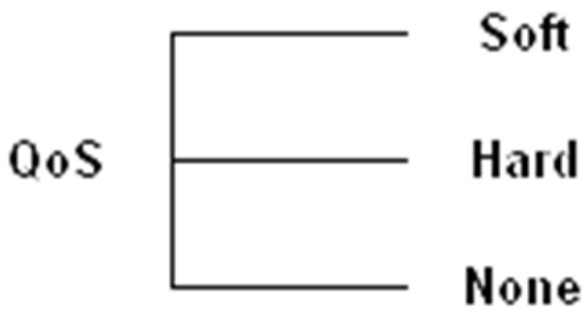



Figure 3.1: Quality of Service Nomenclature [KKM01]

- Binding SLA (BSLA): Application of a resource to a task *e.g.* binding network bandwidth to a socket or a number of nodes to a parallel job. BSLA associated a task defined by a TSLA to a RSLA.

### 3.2.3 QoS Classification

The problem of providing QoS in an efficient manner is a complex issue, involving multiple inter-related aspects, including QoS specification, QoS mapping, resource provisioning, discovery, call admission control, traffic policing, routing, scheduling, and pricing [DKT07]. To overcome these aspects various gurantees for QoS are presented in the form of classification. The three aspects for this are:

- Hard (Strict) QoS: The service providers that are incharge of delivering gurantee, offers a certain specific QoS profile for a given time duration on a contractual basis to the service consumers involved in the corresponding agreement. This mechanism is service specific (discovery, scheduling *etc.*) and depends on the type of agreement established. A safety-critical application, such as remote surgery may require a hard level of availability.

- Soft (Loose) QoS: The provisioning of loose guarantee consists of the capability of clients, to select service instances (with respect to scheduling; time sharing

& priority and with respect to discovery agent and query) that can potentially provide a best effort QoS profile while meeting the clients requirements. This is based on previous measurements of the services on similar machines. From these previous measurements it may be possible to estimate performance values for the service. Loose guarantees are delivered on a best-effort basis, and for this reason, they generally do not require the prior establishment of a Service Level Agreement (SLA) and the consequent negotiation of a contract. Loose guarantees are suitable to those application that can perform adaptation and self-healing operations to cope with requirements that are not met in practice [LG07].

- None QoS: It is independent of both the hard and the soft and this nomenclature will not add any additional attributes to QoS.

Based upon the above description of QoS Specifications, SLA and QoS Classification, We have been identified the most significant parameters for the Resource Discovery process are: Time, Cost, Reliability and Security. Description of all these parameters are given below. Utility and applications of these parameters has been demonstrated in the case study in Section 3.4.

- Time: Time means the duration between the arrival of service request and the completion of the requested service. Time is an important factor to enhance the performance of the Grid. It includes:
  (i) Time needed for communication (between the resource and application) which is communication time
  (ii)Time needed to complete the individual task which is computation time.

- Cost: Cost represents the cost associated with the execution of the Grid tasks. It represents how much a user has to pay for the requested resource or the service. With respect to QoS in Resource Discovery, It consists of three parts.

(*i*) Communication Cost

(*ii*) Computation Cost

(*iii*) Penalties

- Reliability: In terms of QoS, Reliability is the quality of measurements (availability, accessibility, Usage) *etc.*

- Security: Security is a multidimensional issue. Security requirements are usually nonnegotiable and can be explained at various levels. In this work, with reference to QoS in Resource Discovery, only single level security has been considered *i.e.* is Authentication.

## 3.3 Analysis of QoS Parameters in Resource Discovery

Till now very less efforts have been made to attain the efficiency in QoS aware Resource Discovery. A popular example of QoS functions for advance resource reservation and co-allocation is GARA (Globus Architecture for Reservation and Allocation), which provides a support to QoS functions namely discovery, selection and allocation [IF99].

Resource Discovery starts with identification of QoS parameters that meet or exceed the current job requirements. As discussed in Section 3.2.3 the identified parameters of QoS are Time, Cost, Reliability and Security. Resource Discovery schemes generally maintain and query a resource status database. Dissemination of the resource status information is one of the key operations required to keep the resource status databases consistent [Kan03]. Most of the systems follow the following three steps for discovering the resources.

(*i*) Authorization filtering.

(*ii*) Job requirement knowledge.

(*iii*) Filtering to meet the minimal job requirements.

First step namely, Authorization Filtering, deals with issues like who can access which resources/services and under what conditions (at what Time and Cost). It is generally assumed that a user will know which resources he or she has to access in terms of basic services [QZ05]. For this authorization of user is needed. It is checked during authorization process who is going to interact with the Grid resources. An authorization system can be defined as a system that 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 conformation to established policies. In order to understand the architecture well, following elements have been described in details [SB07].

**Subject (SU):** Subject is an entity that wants to access services/resources. It can be a user, a service or any other entity on behalf of user/service.

**Service (SR):** Service is a piece of software that provides some functionality and can be accessed by Subjects or other Services. Services are exposed in the environment and are found by Subjects.

**Resource (R):** Resource is an object that is accessed by Subjects. It can be a CPU, a storage device, software, data, scientific instrument or any other peripheral. Subjects access Resources through Services. In other words, a Resource is a Service.

**Domain (DO):** Domain refers to the set of Subjects and Services under a unique Domain Policy (DP). DO = (SU, SR, DP). The Services in a Domain are provided

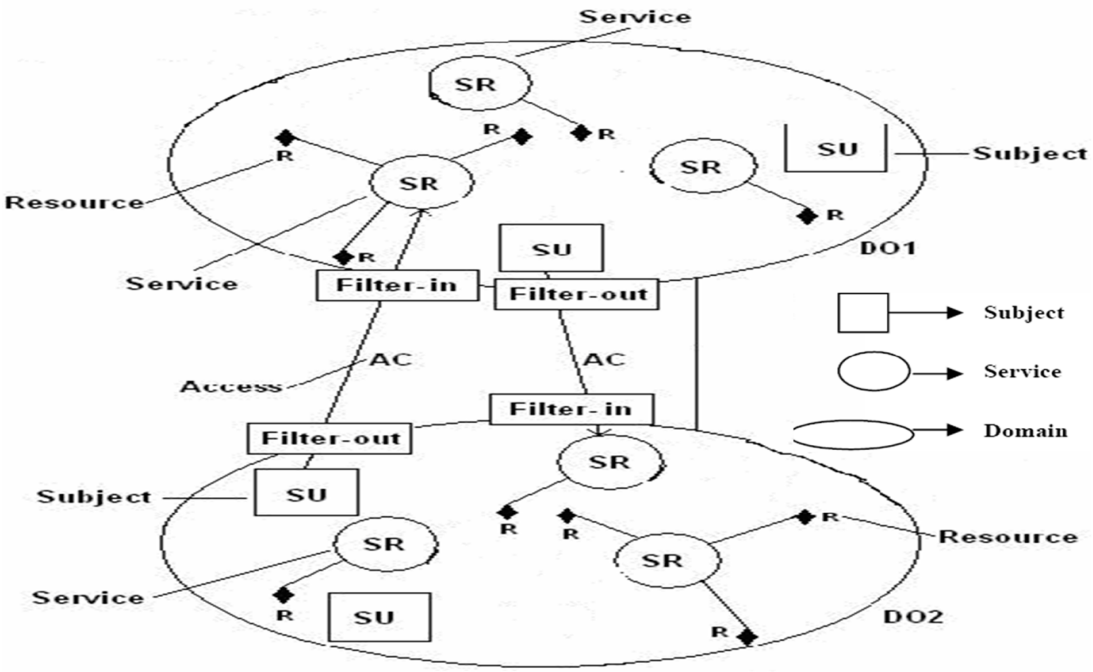



Figure 3.2: Schematic view of Authorization Architecture

by Service Providers and they may belong to same or different physical organizations/institutions. DP refers to the set of rules/regulations/requirements that an entity must conform to in order to be in that Domain.

**Access (AC):** Access is an operation that a Subject performs on Resource/Service. Access operation is represented as AC (SU, SR) *i.e.* Subject (SU) can Access (AC) Service (SR) if it conforms to any Policy.

**Filter:** The rights/privileges of a Subject are different in different Domains. Filter is a component through which rights/privileges of a Subject are filtered for a particular Domain. There are two types of filters (filter-in and filter-out), which has been explained later.

Figure 3.2 shows the schematic view of authorization filtering. This Environment consists of two Domains (DO1 and DO2) along with other elements of the framework. In the diagram Squares(□) represent Subjects, Diamonds (⋄) represent Resources, Rectangles (□) represent Filters and Ellipses (◯) represent Domains. The QoS factors reduce the time taken for checking the authenticity and giving access for the resource usage. At the end of this step the user will have a list of machines or resources to which he or she has access.

When the user has the list of resources as per his or her requirements after accessing the resources available, he or she can finalize the *cost* for a particular period of *time*.

Second step deals with the job requirement knowledge, in which, the user must be able to specify some minimal set of job requirements to further filter out the set of feasible resources.

Third step is to filter out the resources that do not meet the minimal job requirements. The user generally performs this step by going through the list of resources and eliminating those resources that do not meet the job requirements. To achieve this functionality, *Filter* component is used. It filters the rights/privileges of a Subject for a particular Domain. The effectiveness of Filter to provide exact information as per user requirements decides *reliability* of Resource Discovery process. There are two types of Filters: *Filter-in* and *Filter-out*. With Filter-in, the Subject enters the organization with access rights that the target Domain grants to the parent Domain. With Filter-out, the Subject will leave the Domain with access rights that its parent Domain grants it. In other words, the Subject gets the intersection of the rights that its parent Domain grants to it and the rights that target Domain grants to parent Domain.

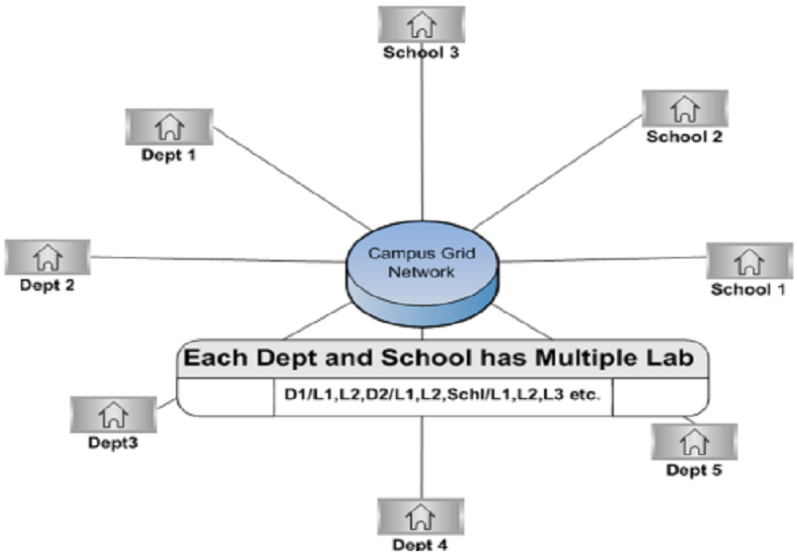



Figure 3.3: Structure of different units in a Campus Grid

## 3.4 Case Study for QoS aware Resource discovery

To elaborate the QoS aware Resource Discovery in Grid environment a case study of an institute's campus has been taken. Different units of the institute namely Dept1, Dept2, Security Center, Center1, Center2, Lab1, Lab2, School1, School2 as shown in Figure 3.3 and have been taken. QoS parameters namely Time, Cost, Reliability and Security have been considered.

Apart from these parameters, various stakeholder in the system are: Users, Resources and Manager:

**Users:** It is an agency that submits the jobs to be executed using the resources in the Grid.

**Resources:** Resources could be clusters of computers, networks, memory space or storage capacity, data repositories, files, attaches peripheral devices and so forth.

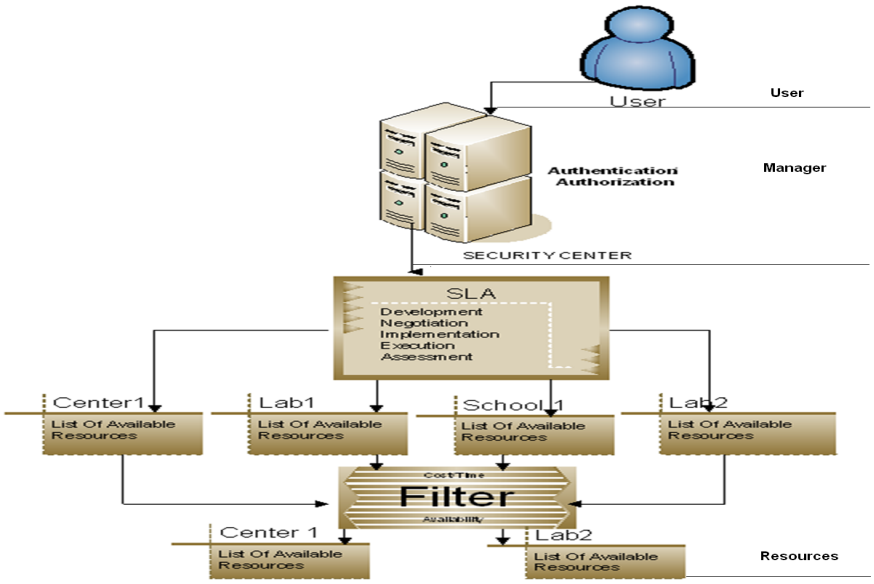



FIGURE 3.4: QoS based Resource Discovery

**Manager:** It is the central authority that manages all the resources and jobs information.

When a user sends request for availability of a particular resource with proper specifications, Manager responses for the availability of the requested resources. If the requested resource is available with the Manager, along with the information of availability of requested resource, it sends SLA to the user. Then, the user fills the SLA and submits to the Manager. The selection of resources is made on the basis of QoS parameters defined in SLA. Only registered users can interact with the Grid. The work of authorization is done by the Central Manager (CM), in this case study. Other works that CM performs are: Managing the SLAs, maintaining information about the resources available, dynamically updating the resource status and above all assigning the most suitable resources for the job submitted.

After authenticating the user, the central manager at Security Center takes the

information from the appropriate SLA. After studying the various parameters which user has demanded, the manager checks for the available resources. In Figure 3.4, the Security Center person finds Center1, Lab1, School1 and Lab2 having some resources free that can be used for the execution of the job. The next stage is of Filtering. In this stage, on the basis of filter component and various QoS parameters, selected resources are further filtered out and only the Center1 and Lab2 remains available. After getting the appropriate resources, the job is submitted to them. The result is again sent back to the manager. In this process, Manager keeps check over the proper execution of jobs, information about SLAs and manages the resource status.

## 3.5 Proposed Design and Implementation Description

Figure 3.4 shows that, if a user has a fully authentication rights and a SLA (CA certificate) then the user is able to reach in any unit of the institute to access the resources that he needs. Resource description file is needs to be designed to describes the description of resources and job types in a unit as shown in Figure 3.5. The various components and technologies proposed for implementations are as shown in Figure 3.6. Where as Table 3.1 & Table 3.2 shows the detailed information that have been required during implementation process. Prototype implementation can be done with the Microsoft Visual Studio 2005 and mySQL Server. Additionally for more attractiveness different visualization tools can be used *e.g.* Java Servelets and CGI. The current case study can be deployed on Grid using GT4 infrastructure which is freely available (Open Source) with database as mySQL. QoS aware Resource Discovery in real life faces some problems while achieving the user defined set of objectives. For example the user may ask for a time which is just not feasible. In second situation time is feasible but cost is not feasible. Practically if database

```
<start job>
<gridid> ``c:\QoS parameters integrated Resource Discovery" </gridid>
<startline>102032432</startline>
<mem>0</mem>
<input> 0 </input>
 <output> 0 </output>
 <jobtype> Primary</jobtype>
 <vmem> 8383 </vmem>
<endtime> 1020323496 </endtime>
</start job>
```

Figure 3.5: Typical Resource Description File

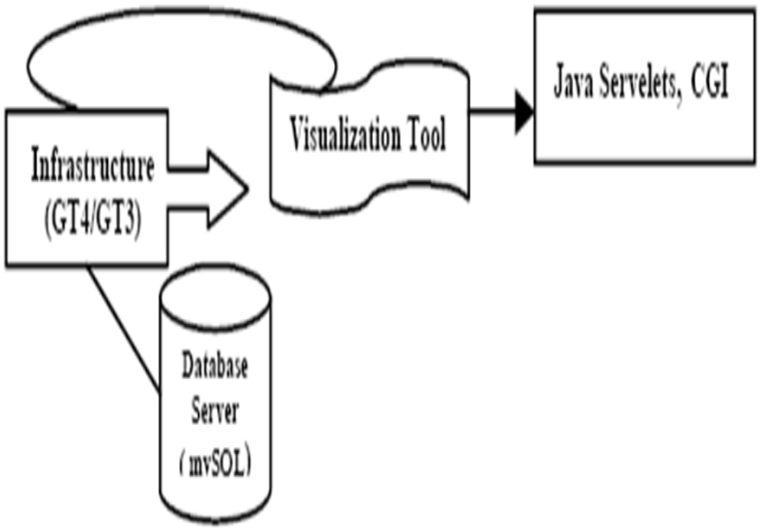



Figure 3.6: Components of Case Study

Table 3.1: Major Information Required for Implementation

| *Type of Information* | *Information* | *Source of Information* |
|---|---|---|
| *Consumer/Provider Organization* | Name, Certificate Authority | CA Certificate |
| *Grid Users* | Certificate Authority, User Name, Organization | CA Certificate |
| *Resources* | Consumer, Resource Provider, Job Specific Information | Database |
| *Grid Jobs* | Consumer, Resource Provider, Job Specific Information | Job Log File |

schema crashes at an instance of time! May be in such cases different resource recovery softwares can be used, but it consumes a lot of time and interrupts the user services. In case of multiple users requesting access to the same resource(s) at same instance of time, some sort of synchronization mechanism is required.

## 3.6 Conclusions

In this chapter, different QoS aware Resource Discovery parameters are identified and analyzed as: Time, Cost, Reliability and Security. Time is responsible for delivery period of a resource. Cost is the amount payable for the delivery of a resource or a set of resources required. Also, penalty cost is included in case of delay in delivery of the resources required. Communication and computation costs are other costs. Reliability is about making delivery sure for a resource or a set of resources required. Security is an issue for ensuring the trust while sharing or exchanging a

Table 3.2: Database Tables

| *Tables* | *Contained Information* |
|---|---|
| *Departments* | List of departments participating in the Grid. Each department may have number of resources contributed to the Grid and number of Grid users. |
| *Resources* | List of local resources of virtual organizations participating in the campus Grid. Each local resource has a list of local users that are mapped by the Grid User. |
| *Grid User ID* | List of Grid User's ID belonging to virtual organization. A Grid User's ID belongs to unique organization. |
| *Grid Job* | List of jobs submitted by a Grid User to local resource. The Grid job is executed under the name of local user associated to the Grid user in the map entry table. |
| *Time* | Time Map strategy which shows the duration of taking the resources. |

resource among users or users and manager.

This chapter achieves first objective of the work and gives insight into the different QoS parameters for efficient Resource Discovery process. In the Next Chapter three Resource Discovery algorithms have been proposed, designed and developed based on different assumptions and scenarios to achieve efficient Resource Discovery in Grid environments.

# 4

# Proposed Resource Discovery Algorithms

*This chapter presents proposed Resource Discovery algorithms and their implementation details. Three Resource Discovery algorithms have been proposed, designed and developed based on different assumptions and scenarios. These algorithms improves upon certain parameters in the existing algorithms like time, costs, matching scenarios, reliability and security. All the algorithms have been implemented and tested for their efficiency. Organization of the chapter is as follows. Section 4.1 deals with the first proposed algorithm i.e."Path Optimization Algorithm". Forthcoming subsections of this section explains the algorithm description and implementation*

*details. Section 4.2 presents the second proposed algorithm i.e."Weight Optimization Algorithm". Further it also explains the algorithm description and implementation details. Section 4.3 explains the parallel versions of both these proposed algorithms. Section 4.4 deals with the third proposed algorithm i.e."Request Matching Algorithm" and forthcoming sections discuss the algorithm description, implementation details and different scenarios for designing the algorithm. Additionally, section 4.5 presents the comparative analysis of these algorithms.*

Resource Discovery deals with finding, locating, searching and organizing the matching data (information, facts, figures, numbers *etc.*), in response to queries of a user or an automated mechanism. Therefore, in large distributed network environments like Grid environment, it needs to be very efficient and optimized for a set of parameters. Resource Discovery with following conditions has been considered in this work:

(*i*) Execution in shortest amount of time.

(*ii*) Execution with special access policy and given standards.

(*iii*) Matching of the available resources.

(*iv*) Candidature of Resources.

Keeping in view of these complex issues in Resource Discovery process, graph theoretic approach has been considered for designing first two algorithms as explained below:

## 4.1 Path Optimization Algorithm

The first proposed algorithm is "Path Optimization Algorithm". This Algorithm is based on the concept of "connected graphs" and their trees. As the name suggests,

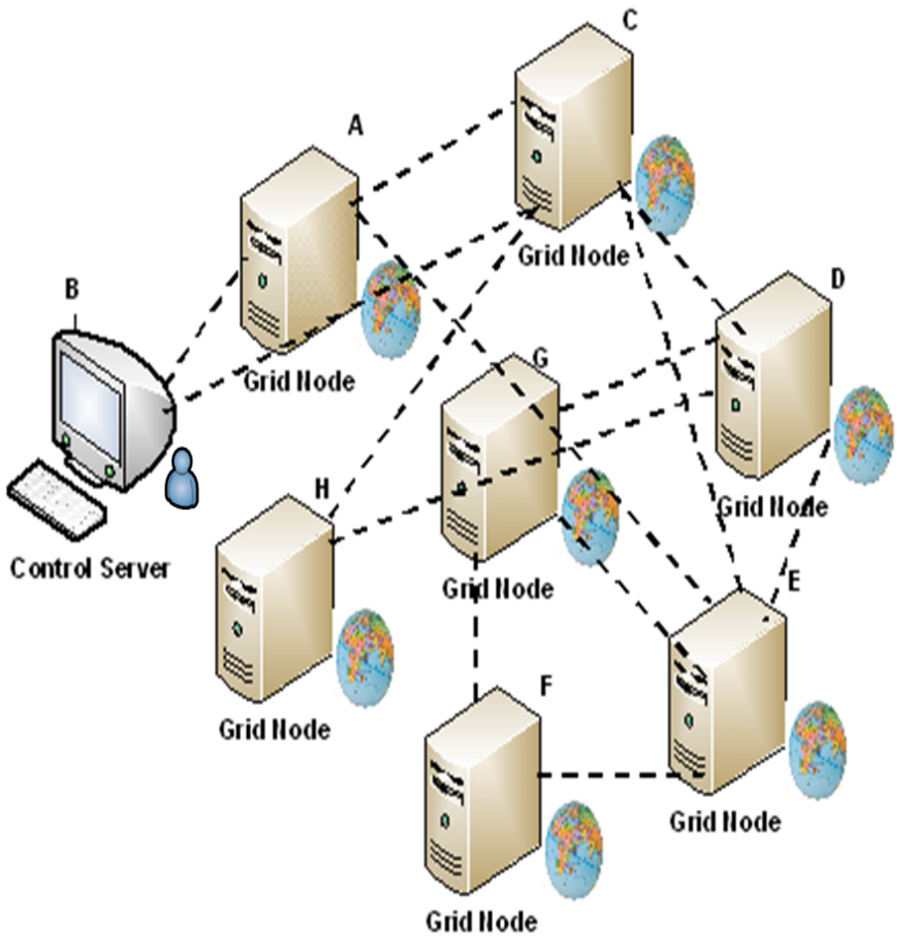



Figure 4.1: Schematic View of Grid with similar Resources

this algorithm tries to optimize the path traversed for Resource Discovery. The objective of this proposed algorithm based on graph theoretic approach is to design a connected graph of eight nodes, divided into tree and discover the given resource in $\mathcal{O}(n^3)$ time. As shown in Figure 4.1, Grid resources are initially mapped as connected graph. Figure 4.2(A) shows the corresponding graph of the Grid shown in Figure 4.1 and Figure 4.2(B) shows its tree. This algorithm finds the resource by traversing a shortest path and also minimizes the number of hops (steps) to reach on the leaf node (F, G, D).

The advantages of converting the connected graph into its trees are as follows:

($A$) Tree is minimally connected.

($B$) There is one and only one path between the root node to the leaf node.

*Terminology used to describe the graph is given below:*

**Graph:** A Graph G=(V,E) consists of set of objects $V = \{v1, v2,...\}$ called vertices, and another set $E = \{e1, e2,...\}$, whose elements called edges, such that each edge $e_k$ is defined with an unordered pair $(v_i, v_j)$ of vertices.

**Walk:** A walk is defined as an alternating sequence of vertices and edges, beginning and ending with vertices, such that each edge is incident with the vertices preceding and following it. No edge appears (is traversed) more then once in a walk. For example in Figure 4.2(A) A→ B→ C→ H is a walk.

**Path:** An open walk in which no vertex appears more than once is called a path. For example in Figure 4.2(A) A→B→C→D is a path [Deo02].

**Connected Graph:** A connected graph $G$ is defined as a simple graph in which a vertex is joined to every other vertex exactly by one edge or there is at least one path between every pair of vertices in $G$. For example Figure 4.2(A) shows a connected graph.

**Tree:** A tree is collection of elements called nodes, one of which is distinguished as a root, along with a relation ("parenthood") that places hierarchical structure on the nodes. Formally, a tree can be defined recursively in the following manner. Figure 4.2(B) shows a tree.

- A tree is a connected graph without any circuits.

- A single node by itself is a tree. This node is also the root of the tree.

*Some properties of the trees that are needed to be specifically mentioned here are:*

- There is one and only one path between every pair of vertices in a tree $T$.
- If in a graph $G$ there is one and only one path between every pair of vertices, then $G$ is a tree.
- A tree with $n$ vertices has $n-1$ edges or any connected graph with $n$ vertices and $n-1$ edges is a tree.
- A graph is a tree if and only if it is minimally connected.

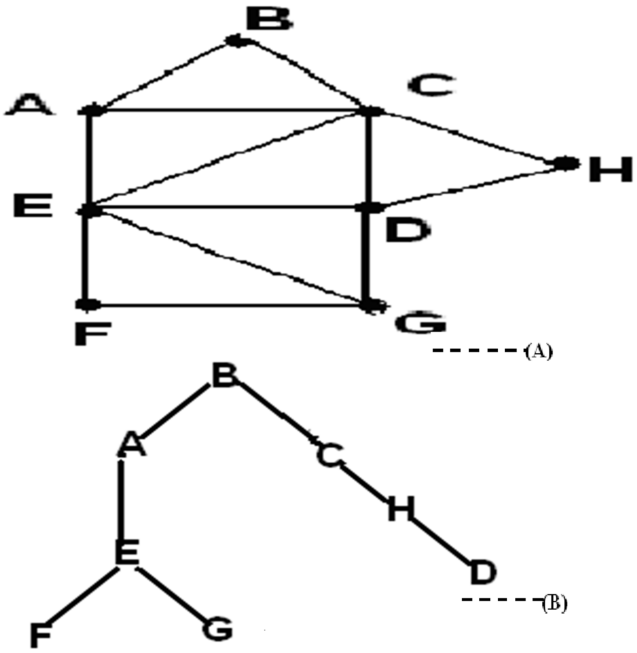



Figure 4.2: Connected Graph and its Tree

### 4.1.1 Algorithm Description

The proposed algorithm has following sequence of steps that can be performed to find the resource in smallest unit of time:

**Step I:** Compute the Adjacency Matrix of a given graph.

*Step A:* place 1 if there exists an edge between $v_i$ and $v_j$.

*Step B*: otherwise place 0.

**Step II:** Calculate the Minimum path length between the nodes using Warshall's Algorithm (best available serial algorithm)

```
for (k := 1 to n)
for (i := 1 to n)
if Adj[k,i] then
for (j := 1 to n)
Adj[i][j] := (Adj[i][j] or Adj[k][j]).
```

### 4.1.2 Implementation Details

A heterogeneous computing environment has been created using Pentium-I, Pentium-II, Celeron, Pentium-III and Pentium-4 machine running Windows, Linux and Solaris Operating System. The "*Path Optimization Algorithm*" is executed by computing the shortest path of the connected graph, obtained from the Grid (Figure 4.1) under considerations. Adjacency Matrix of the graph, as shown in Figure 4.3 obtained using the steps described above and shortest path is calculated.
Different Data Structure used for the implementations are:

```
struct path {/*Declare path as a structure*/} {
predecessor; {/*a node*/}
distance; {/*minimum distance of the node from the source*/}
};{/*End of Structure*/}
Adjancymatrix[MAX][MAX] {/*The Adjacency Matrix of the graph*/}
pathmatrix[i][j] {/*To find the path*/}
```

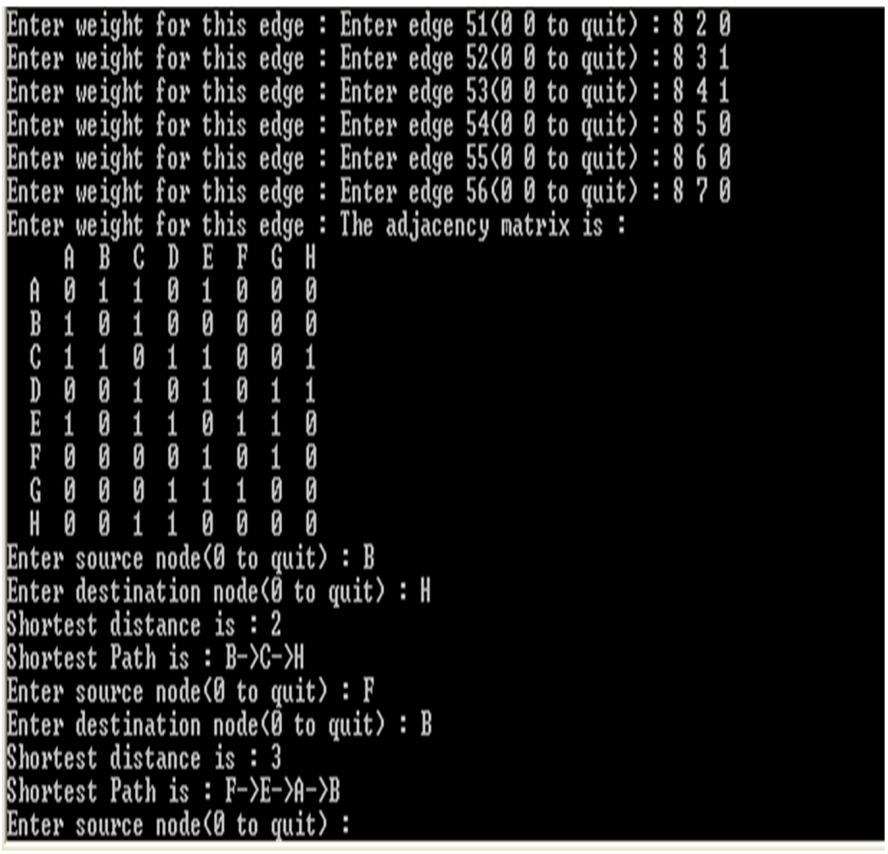



Figure 4.3: Running Path Optimization Algorithm for graph shown in Figure 4.2

{/*Functions used for finding the shortest path:*/}

creategraph( );

displaygraph( );

findpath( );

The proposed algorithm has been tested successfully and results obtained have been demonstrated on TUGrid testbed in 'Chapter 5'.

### 4.1.3 Complexity Analysis

The main emphasis in "Path Optimization Algorithm" is to propose an efficient algorithm. Step I, which computes the Adjacency Matrix of the graph has two **for**

loops so the complexity $\mathcal{O}(n^2)$. Step II uses three **for** loops and computing the Transitive Closure of Adjacency Matrix thus the Complexity is $\mathcal{O}(n^3)$

Thus, total complexity is $\mathcal{O}(n^2) + \mathcal{O}(n^3) = \mathcal{O}(n^3)$

## 4.2 Weight Optimization Algorithm

The second proposed algorithm is "Weight Optimization Algorithm". This algorithm is based on the concept of the "directed weighted graph" and it optimizes the total cost/time using its Minimum Spanning Tree (MST). This Resource Discovery algorithm works with assumptions that there are number of heterogeneous resources (vertices) available, each having their independent costs of usage per unit time. The cost of the resources consists of two components computational cost and communication costs. Computational cost ($comp_i$) shown along with the nodes which is the cost of computation per unit time assigned with the resource. While communication cost ($comm_{ij}$) shown on the edges is the cost of communication per unit time, from the predecessor node to successor node.

These costs are generally known before hand as shown by an example in Figure 4.4. Heterogeneous resources available along with their access path and respective costs are mapped into a directed weighted graph as shown in Figure 4.5(A).

*Terminology used to describe the directed graph is given below:*

**Vertices:** Vertices ($V$) are the set of nodes in a Graph. As shown in Figure 4.4, $V = \{D1, PD1, LP1, S1, PC1, P1, S2, MF1\}$ are the vertices.

**Edges:** The lines connecting the vertices are known as edges (E). As shown in Figure

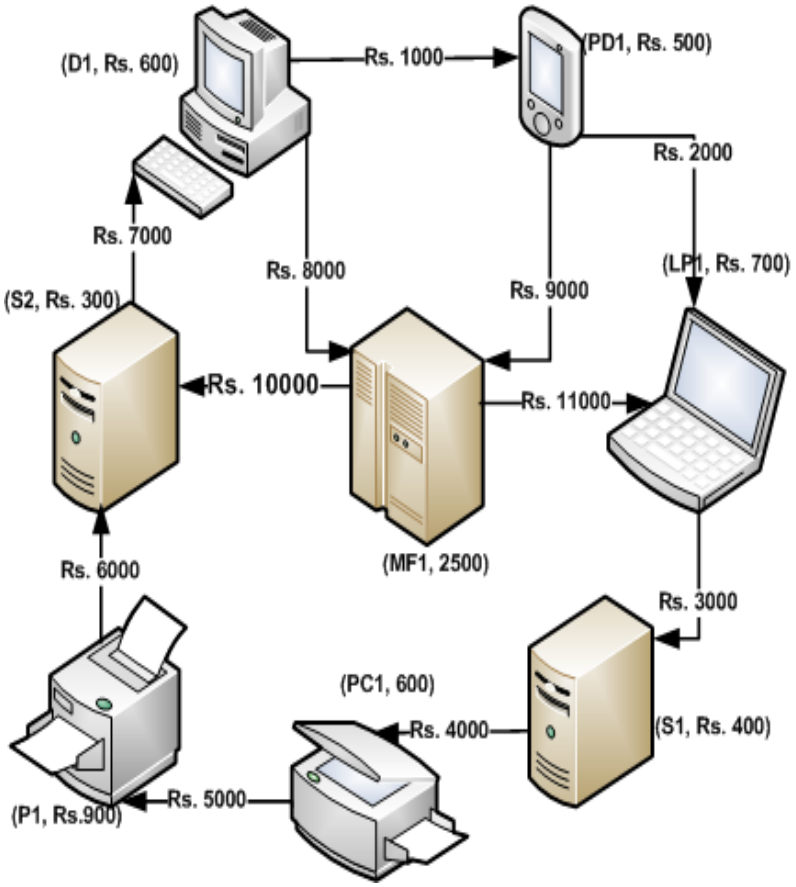



Figure 4.4: A Heterogeneous Grid

4.4, $E = \{(D1, PD1), (PD1, LP1), (S1, LP1), (PC1, S1)(P1, PC1), (S2, P1), (D1, MF1)\}$ *etc.* are the edges.

**A directed graph:** A directed weighted graph (diagraph or $DG$) consists of set of vertices $V$, a set of edges $E$, and a mapping that maps every edge onto some ordered pair of vertices ($E_{D1,PD1} \| D1 \rightarrow PD1$). A diagraph is also referred to as an oriented graph. A Grid as shown in Figure 4.4, has been mapped as directed weighted graph as shown in Figure 4.5(A).

### 4.2.1 Algorithm Description

The proposed algorithm has following sequence of steps:

**Step I:** From a directed weighted graph select any vertex, having the least computation cost as the current node to start.

{/* Select any vertex at random if there are number of vertices with same minimum computational costs. Let the node be $V_i$ and $T$ is blank tree.*/}

**Step II:** Find the nearest neighbour of the current node having minimum edge weight. Add edge $E_{ij}$ into the tree T. Let the node be $v_j$

{/* Select any edge at random if there are more than one edges with the same minimum edge weight. Let the edge be $E_{ij}$ and make the $V_j$ to be current node.*/}

**Step III:** Repeat Step II until all the vertices are visited.

**Step IV:** The tree T so obtained is the MST.

**Step V:** Compute the total cost of using the resources by adding all $comp_i$ and all $comm_{ij}$ that fall on the path of the MST.

Currently, assumption is that it takes unit time for computation within a node and also for communication from one node to adjacent node. In case we remove this assumption and consider the resources $V_i$ has been used for time $T_i$ units and it takes $T_{ij}$ time to reach from node $i$ to node $j$. Then the total cost would be the weighted sum, *i.e.* computational costs will be $T_i * comp_i$ and respective communication cost will be $T_{ij} * comm_{ij}$.

### 4.2.2 Implementation Details

A heterogeneous computing environment has been created using Pentium-I, Pentium-II, Celeron, Pentium-III and Pentium-4 machines running Windows, Linux and Solaris Operating Systems. Implementation has been done using Microsoft Visual Studio 2005 as the front-end and MySQL Server as the back-end. *"Weight Optimization Algorithm"* is executed by obtaining the MST of the directed weighted graph obtained from the heterogeneous Grid under considerations. There exist many standard algorithms for constructing the MST [Raj98] like Prim's Algorithms, Kruskal's Algorithm *etc.* [AVA04], [AVA03], [SB03]. The algorithm proposed here is mainly based on the Prim's algorithm however, it has been modified to have the scope of parallelization, as explain in the next section.

Different Data Structures that are used are:
W[MAX][MAX]{/*Initialize the Weight Matrix*/}
*root* {/*an arbitrary node chosen as root*/}
$T_c$[root] = INFINITY {/*Initially Total cost is set to INFINITY*/}
$T_c$[node] = W($comp_i$ + $comm_{ij}$) {/*Compute the total cost by adding the $comp_i$ and $comm_{ij}$*/}

The critical steps in this algorithm is method for efficient determination of the "closest" node to a partial spanning tree. Initially, when the partial tree consists of a single root node, the Total Cost ($T_c$) of any other node from the tree, $T_c$[node], is equal to the weight($comp_i$+$comm_{ij}$). When a new node, $comp_i$, is added to the tree $T_c$[node] is modified to the minimum of the $T_c$[node] and weight($comp_i$+$comm_{ij}$). The node added to the tree at each point is the node having the lowest weight. An additional array closest[node] points to the node in the tree such that $T_c$[node] = weight(closest[node], node); that is the node in the tree closest to node.

*// From a directed weighted graph select a vertex having the least cost as a root node to start.*

*root* = an arbitrary node chosen as *root*;

***for*** (every root node in the directed weighted graph) {

$T_c$[node] = weight($comp_i$+$comm_{ij}$)

closest[node]=*root*;

} /* end for */

$T_c$[root] = INFINITY;

$comp_i$ = *root*;

**for** (i=1; i < number of nodes in the graph; ++i) {

/* find the node closest to the tree*/

*minimumcost* = INFINITY;

**for** (every node in the directed weighted graph)

**if** ($T_c$[node] < *minimumcost* {

*minimumcost* = $T_c$[node];

}/* end if */

/*add the closest node to the tree*/

/*and adjust weights*/

addnode (closest[$comp_i$], $comp_i$);

$T_c$[$comp_i$] = INFINITY;

for (every node adjacent to $comp_i$)

**if**(($T_c$ of the[node] < INFINITY

&& (weight($comp_i$+$comm_{ij}$) < $T_c$[node])) {

$T_c$[node] = weight($comp_i$+$comm_{ij}$); {

closest[node]=$comp_i$;

}/* end if */

} /* end for */

The detail description of the MST that has been obtained after the implementation of the given directed graph is:

| Attached Vertices | Edges | Total Cost (Rs.) |
|---|---|---|
| D1 | {D1,PD1} | 1000+500=1500 |
| PD1 | {PD1,LP1} | 2000+500=2400 |
| LP1 | {LP1,S1} | 3000+700=3700 |
| S1 | {S1,PC1} | 4000+600=4600 |
| PC1 | {PC1,P1} | 5000+700=5700 |
| P1 | {P1,S2} | 3000+700=3700 |
| S2 | {D1,MF1} | 8000+2500=10500 |

### 4.2.3 Complexity Analysis

The main emphasis in "Weight Optimization Algorithm" is to propose an efficient algorithm to optimizes the cost using its MST. Analysis of the proposed algorithm has been done. As **for** loop is executed twice to select the *closest node* with *minimumcost*. So, the complexity is $\mathcal{O}(n^2)$.

In the proposed algorithm, computational cost ($comp_i$) and communication costs ($comm_{ij}$) are added to select the efficient resource and MST (as shown in Figure 4.5(B)) is obtained which ensures the minimum $comp_i$ and $comm_{ij}$. This algorithm has been tested successfully and the results obtained have been demonstrated in TU

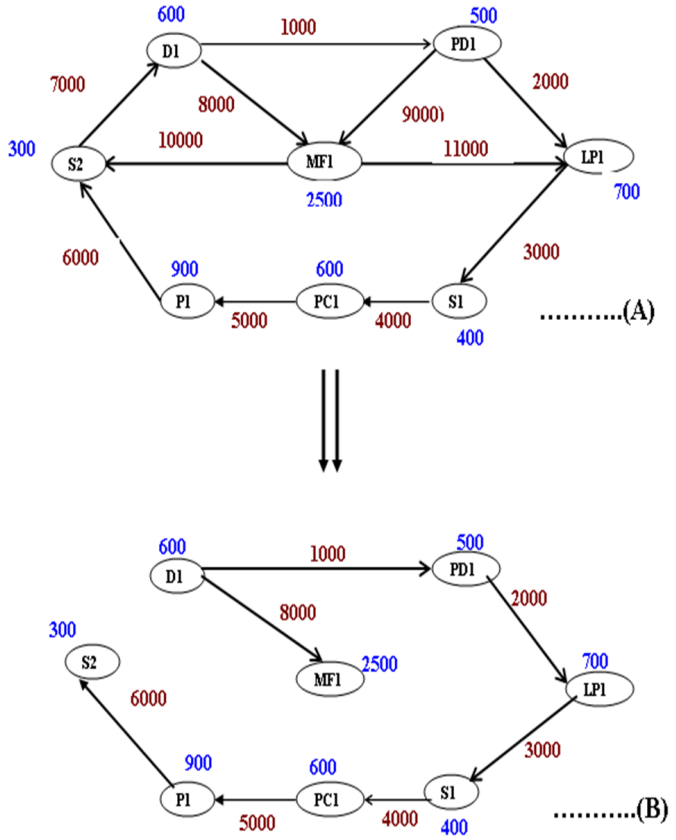



FIGURE 4.5: Minimum Spanning Tree of Directed weighted Grid

Grid testbed, reported in the next chapter. However, the next section explains how both these algorithm can be made more efficient.

## 4.3 Parallel Versions of the proposed Algorithms

Both the proposed algorithms *i.e. Path Optimization Algorithm* and *Weight Optimization Algorithm,* have scope of parallelization. In both these algorithms, work to find an optimal path between nodes with given constraints, *i.e.* computing Transitive Closure of given adjacency matrix. Also, anytime a new node is added or withdrawn (resources may leave or join at any time in Grid environment), this process of computing Transitive Closure is repeated again. Therefore, both these

algorithms can be efficiently parallelized by computing the Transitive Closure in parallel. Various parallel algorithms exists for parallelizing the Transitive Closure Computation. These algorithms utilize the number of processors that are available in any parallel, distributed and Grid environment.

One method of parallel Transitive Closure algorithm [GKS02] mainly emphasizes on how to minimize the idle time of each processor in the most appropriate manner. The distribution to different processor of rows of an adjacency matrix for which transitive closure is to be computed in parallel is based on the Weightage Per Processor (WPP). This algorithm has been targeted to a network of machines where each machine is assumed to have a considerable amount of computational power, so it is most appropriate to assign more than one row to each machine rather than assigning single row to each machine. Using this algorithm, one can not only drastically reduce the total Transitive Closure Computation time but also optimally utilized the combined computational power available with the network of the machines.

Another method of parallel Transitive Closure Computation algorithms [Baw04] employes the master-slave approach as shown in Figure 4.6. The master processor divides the computational task into the number of subtasks depending upon the available slave processors for execution. The distribution of subtasks in the Transitive Closure Computation is an improvement over row-wise distribution of the matrix, where a greater interaction is required between the master and the slaves, resulting into increased interprocess communication. In row wise distribution, all the $n$ tasks execute concurrently as each slave does computation for one row. This leads to worst case and further can lead to two iterative loops to be executed, leading to the complexity of $\mathcal{O}(n^2)$.

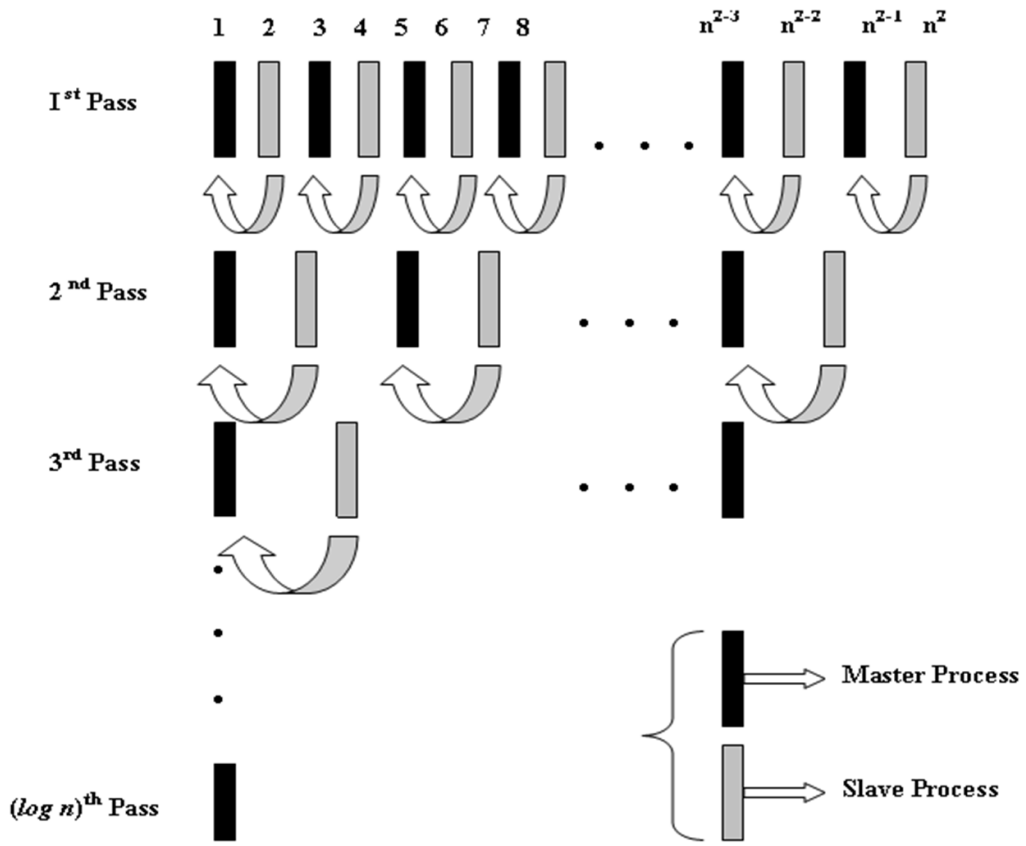



FIGURE 4.6: Master-slave tasks

In current approach master processor spawns $n^2$ tasks, where each task receives a unique row-column set. Each task is a member of a group with unique group identifier and the task with even group identifier is a master and odd group identifier tasks are slaves. However the computation at master and slave processor is different at each stage. As explanation of "Path Optimization Algorithm" has been done in section 4.1 and has the complexity of $\mathcal{O}(n^3)$. While parallel algorithm with row wise distribution has complexity $\mathcal{O}(n^2)$. Where as the proposed algorithm with Master-Slave approach has the complexity of $\mathcal{O}(logn)$ with $n^2$ processor which is more optimal. And the complexity of parallel weight optimization algorithm becomes $(nlogn)$.

## 4.4 Request Matching Algorithm

This section explains the third proposed algorithm *i.e.*"Request Matching Algorithm". The "Request Matching Algorithm" considers the following parameters:

($A$) The number and types of available resources.

($B$) Type of resources requested by the users.

($C$) Number and types of resources allocated to each user.

This algorithm works with three matchmaking principles: first - to find the acceptable matches; second - to superwise and modernize (update) lists of the requesters and providers; third - dynamically updating the data in the resource pool. For finding the acceptable matches, matchmaking principle work through the requester and provider lists to match appropriately. When the match is found corresponding entries are provided to the requester and corresponding resources are marked as "allotted" for that much time in the provider list. In regular interval of time, matchmaker updates the requester list and provide list.

Scope of this algorithm is to find the best resource match from a set of all available matching resources for a given job. Matchmaking deals with the finding of resources that satisfies the enduser's requirements to the maximum [KML05]. Matchmaking is done depending upon whether the enduser request and the resources available with resource provider match or not [UB07]. The match between enduser's and resource provider is made based on the match between capabilities of the resources available and resources requested. Resource Discovery requirements for the Grid services can be generic (simply as RAM, H/W, S/W etc.) and specific (256Mb RAM, HP printer, and specific CASE tools *etc.*). To satisfy these requirements a Grid Resource Discovery Model based on Ontology Approach has been proposed. This Resource Discovery model consists of five layers, each layer performing the distinct task, as explained below:

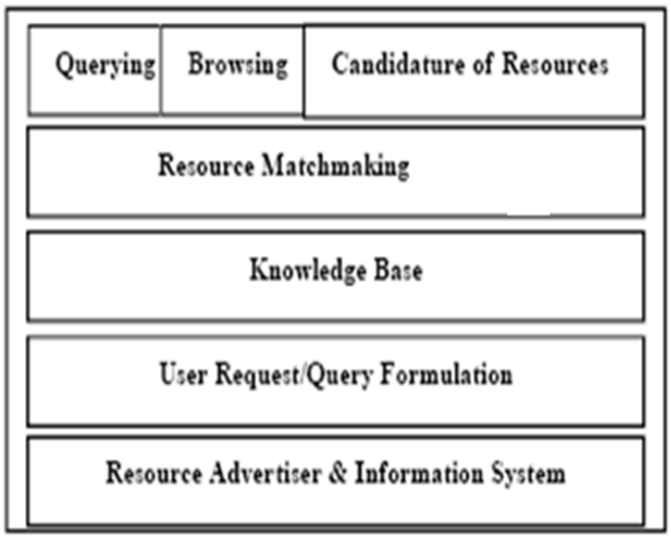



FIGURE 4.7: Proposed Layered Discovery Model for matching based Resource Discovery

*Resource Advertiser and Information System Layer:* In Grid environment, resources are dedicated to applications. This layer consists of collection of heterogeneous resources and how these resources are efficiently utilized, resulting in decreased infrastructure costs, reduced processing time, increased responsiveness and faster time-to-market. In this layer resources may be combined with Agents to produce high-level performance of corresponding resources.

*User Request/Query Formulation Layer:* It needs to be details regarding the resources requested by the user and need to be asked and formulate the particular query. A query is a search of missing/needed information. For example a search query is: Requirements = (Arch== "INTEL" && OpSys == "LINUX") || (Arch == "INTEL" && OpSys == "SOLARIS10") || (Arch == "P4" && OpSys == "WindowsXP"). The content of the query describe the particular resource having certain parameters and helps to search the same.

*Knowledgebase layer*: It contains detail information regarding all the available resources for example, type of resources, availability, utilization of the resources, processing time of the resources, and Ontology of the resources that describes the semantic properties and semantic relations of the resources.

*Resource Matchmaking Layer:* Ontology is a shared understanding of some domain of interest, which is often realized as a set of classes (concepts), relations, functions, axioms and instances. In this layer matching of all the defined attributes of the resources takes place.

*Service layer:* Offers a set of functions related to query, browse and candidature of resources. After this the process of publishing takes place.

The Resource Matching algorithm has been proposed, designed and developed with different scenarios as explained below:

(*i*) Exact Hit Scenario: If requested resources are exactly available at once in the provider list and matching takes place at once. The query specifies the desired values for all resource attributes sought. For example, Architecture='C2D' and CPU-Speed='2.6GHz' and price='20 Grid dollars per second' and RAM='2MB' and No. of processors='5' and Secondary free space='1GB' and Interconnect bandwidth='50MB' and OS='linux'.

(*ii*) Multiple Match Scenario: If end user requests many resources and they are matched at many places. Different tie breakers can be used in this scenario for example: locality of the resources, utilization of the resources and cost of the resources etc. For example, Architecture='P4' and CPU-Speed='1GHz' and RAM='512MB' and No. of processors='3' and Secondary free space='300MB' and Interconnect bandwidth='30MB' and OS='WindowsVista'. This is available at more than one places.

(*iii*) Priority Based Match Scenario: In this case there are many requirements wheatear user assigns priority to every attribute of a resource matching is done based on this priority assigned by the user for example, if a requirement having top priority is not met that particular resource is not considered more. More explanation of this algorithm is described in Section 4.4.1

(*iv*) Miss Scenario: In this scenario requested resources as it is are not available in provider list.

### 4.4.1 Algorithm Description

The advertisement of resources is done to the resource consumers by resource provider. Matching process takes place between the enduser and provider list. There are various possible scenarios as explained above Figure 4.8, user TUGrid5 and user TUGrid7 both request the same type of resource *i.e.* 'RAM' concurrently. Priority of user TUGrid5 is higher then the user TUGrid7. In this case pairing is done with the user TUGrid7 and provider TUGrid17 having the same priority. By using this approach we select a resource provider.

### 4.4.2 Implementation Details

A heterogeneous computing environment has been created using Pentium-I, Pentium-II, Celeron, Pentium-III and Pentium-4 machines running Windows, Linux or Solaris Operating Systems. The *"Request Matching Algorithm"* is executed based on the different scenarios as explained in the above section. The algorithm uses the following data structures:

- AVAILABLE [j] = number of $j_{th}$ resources available.
- MAXIMUM [i][j] = maximum number of $j_{th}$ resources that enduser $i_{th}$ will use.

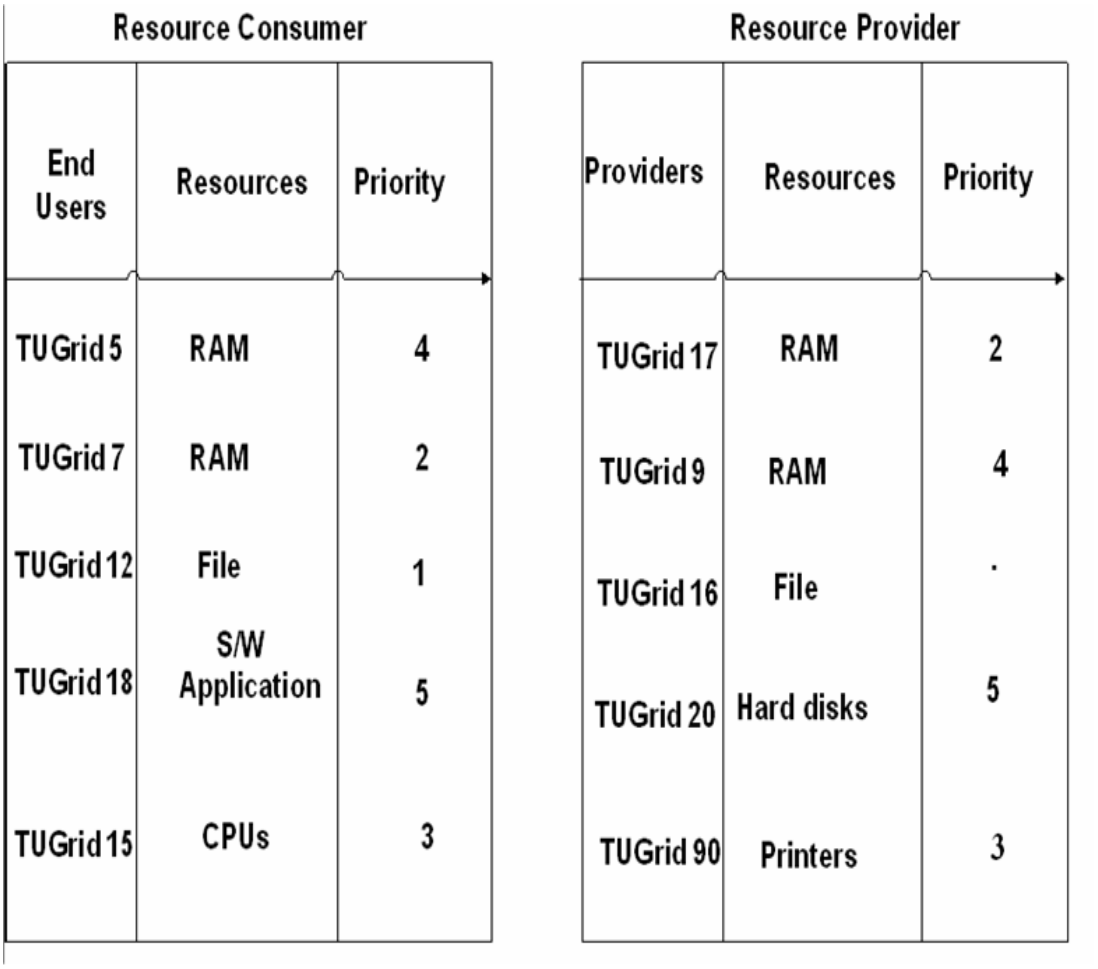



Figure 4.8: Priority based Scenario

- REQUEST [i][j] = number of $j_{th}$ resources that $i_{th}$ enduser request.
- MATCH [i][j] = number of $j_{th}$ resources that match with the $i_{th}$ enduser requirement.
- ALLOCATE [i][j] = number of $j_{th}$ resources that have been allocated to $i_{th}$ enduser .

Structure of database tables that has been used for the implementations are as described below:

Table 4.1: Resource Requester Table

| Resource Information | Type |
|---|---|
| RequesterID | int |
| ResourceID | int |
| Resource Priority | varchar |
| Number of Resources | int |

Table 4.2: Resource Provider Table

| Resource Information | Type |
|---|---|
| ResourceID | int |
| Resource Provider | varchar |
| Resource Status | varchar |
| Cost(GridDoller/second) | varchar |

Table 4.1 is the Resource Requester table that contains the information about the requester fields. Table 4.2 is the Resource Provider Table which contains the resource provider's information. Table 4.3 contains the information about the matching field of the resources. Table 4.4 contains the information about all the resources.

The allotment of various scenarios is explained below:

***Exact Hit Scenario***

If requested resource is matched exactly once with the provider list

{

Allocate the requested resource to the user for the requested time

}

***Multiple Match Scenario:***

Table 4.3: Resource Match Table

| Resource Information | Type |
|---|---|
| ResourceID | int |
| RequesterID | int |
| Number of resources alloted | int |
| Start Time | varchar |
| End Time | varchar |
| Unit Cost | varchar |

Table 4.4: Resource Table

| Resource Information | Type |
|---|---|
| ResourceID | int |
| Processor | varchar |
| RAM | varchar |
| Hard Disk | varchar |
| Operating System | varchar |
| Processor Speed | varchar |

If requested resource is matched at many places in the provider list

{

Report all the matched resources

}

***Priority Based Match Scenario:***

If requested resource is matched at many places in the provider list

{

Check the priority of all the requested resources and based upon priority of the requested user assign the resources

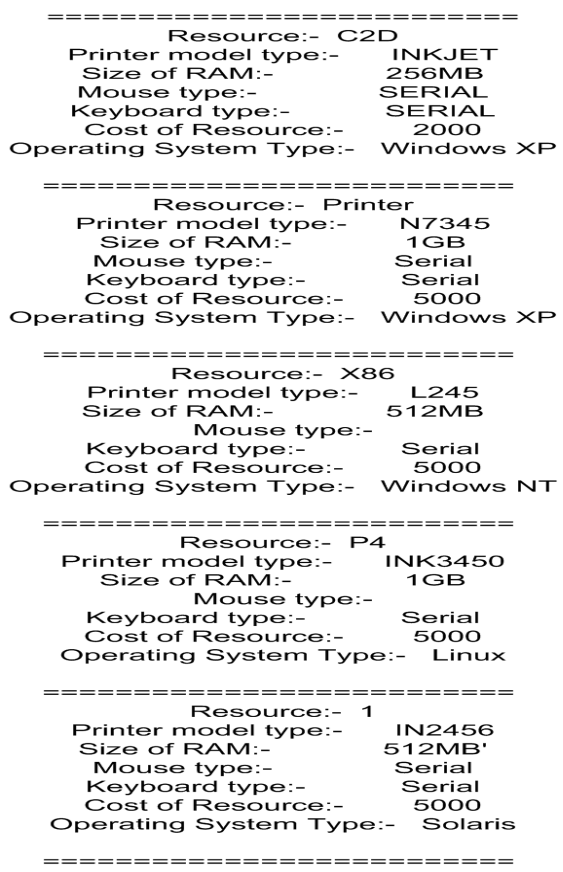



FIGURE 4.9: Details of resource table at particular instance

}

***Miss Scenario:***

If requested resources are not found in the provider list

{

convey the message to the user "requested resources are not available"

}

### 4.4.3 Complexity Analysis

The complexity Analysis is shown in Table 4.5. Table 4.6 shows the Qualitative

Table 4.5: Complexity Analysis of the Request Matching Algorithm

| Matching Scenario | Complexity |
|---|---|
| Exact Hit | $\mathcal{O}(n)$ |
| Multiple Match | $\mathcal{O}(n)$ |
| Priority Based Match | $\mathcal{O}(nlogn)$ |
| Miss Scenario | $\mathcal{O}(n)$ |

comparison of Request Matching Algorithm with the existing middleware, condor matchmakinking mechanism.

## 4.5 Comparative Analysis of the proposed Algorithms

Table 4.7 provides the complexity of various Resource Discovery algorithms with respect to time.

## 4.6 Conclusions

Resource Discovery deals with information retrieval of resources in a fairly standard and efficient manner. In Grid Computing resources are increasing rapidly, efforts are to make the environment more manageable and efficient. This chapter proposes three promising algorithms namely: “Path Optimization Algorithm”, “Weight Optimization Algorithm” and “Request Matching Algorithm”. A comparative analysis with existing algorithms has been done. Attempts have been made to optimize different performance metrics like time, cost, reliability and security. Path

Table 4.6: Qualitative Comparison of Request Matching Algorithm with Condor Matchmaking Mechanism

| ***Propoerties*** | ***Condor*** | ***Request Matching Algorithm*** |
|---|---|---|
| Matchmaking | Matchmaking is a symmetric process | Matchmaking is an asymmetric process |
| Ability to describe Matching Preference | Very much limited due to matching is based on the syntax | Matching preference and ranking criteria easily be added with resource and request |
| Expressiveness | Less Expressive | More expressive due to the asymmetric description, the request can be modeled specifically for domain specific application |
| Implementation | In Condor a daemon running called CEMon which takes the state of the cluster and converts it into a Condor Classad | It is done with the Microsoft Visual Studio 2005 |
| Future Scope | Central to Condor's matchmaking capabilities and future scope is less | Future scope is high as it can be deploy on any Middleware |

Table 4.7: Comparative Analysis of the Algorithms

| Algorithms | Time Complexity |
|---|---|
| Name-Dropper | $\mathcal{O}(log^2)n$ |
| KuttenPeleg | $\mathcal{O}(logn \ \ log^*n)$ $log * n$ is the minimum number of times that the logarithm function is applied to $n$ such that the resulting value is less than or equal to 1. |
| Path Optimization Algorithm | $\mathcal{O}(n^3)$ |
| Weight Optimization Algorithm | $\mathcal{O}(n^2)$ |
| Request Match Algorithm | $\mathcal{O}(nlogn)$ |
| *Parallel Path Optimization Algorithm | $\mathcal{O}(logn)$ with $n^2$ processors |
| *Parallel Weight Optimization Algorithm | $\mathcal{O}(nlogn)$ with $n^2$ processors |

Optimization Algorithm optimizes the path traversed for the Resource Discovery process thus optimizes the time taken in Resource Discovery process. On the other hand, Weight optimization Algorithm is based on the concept of directed weighted graph and it optimizes the total cost/time needed for Resource Discovery process. Request Matching Algorithm helps to find the best resource from a set of all matching resources for a given job, and it works with the three matchmaking principles they are, first - to find the acceptable matches; second - to supervise and modernize lists of the requester and provider; third - dynamically create the data updating in the resource pool for finding the acceptable matches.

Complexity analysis as well as comparative analysis of these algorithms has been

made. Next chapter explains the customized testbed: 'TUGrid testbed' that has been developed as part of this work and demonstrates the usefulness of this work. Different test cases are considered for the evaluation of the proposed algorithms. The testcases and results obtained have been explained in the next chapter.

# 5

# Deployment, Testing and Validation of proposed Algorithms

*Previous chapter described the three proposed algorithms for Resource Discovery process and their implementation. In this chapter Section 5.1 starts with the design and implementation of a testbed named 'TUGrid Testbed' that has been developed and then used to test the proposed Resource Discovery algorithms. Section 5.2 explains the evaluation parameters of three algorithms. Section 5.3 presents various experiments that evaluate the designed algorithms with the functionalities of TUGrid testbed. Different types of QoS parameters including time, costs, reliability and security are also observed and presented. Finally last section summarizes the main*

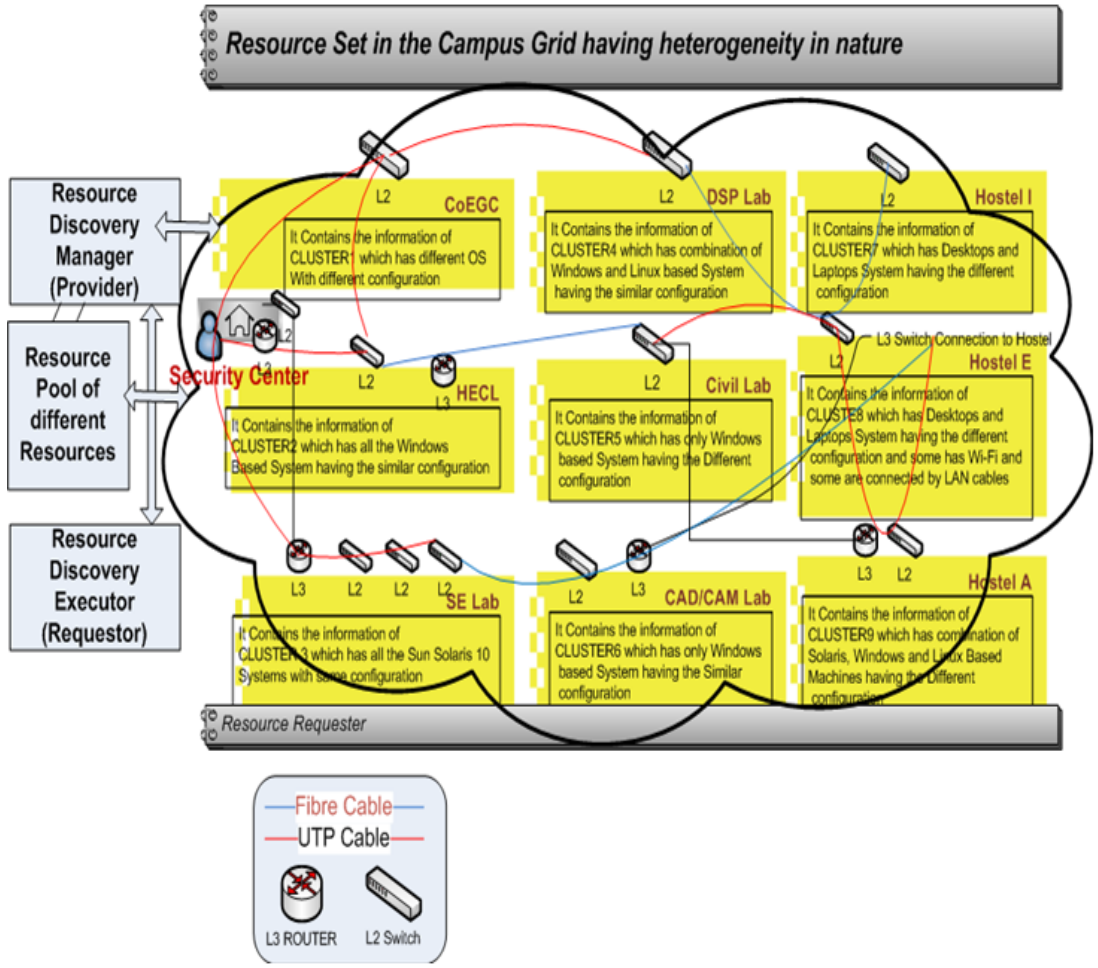



Figure 5.1: Architectural layout of the TUGrid Testbed

*finding of the experiment and presents the analysis and conclusion of this chapter.*

## 5.1 TUGrid Testbed

To test and evaluate the performance of Resource Discovery algorithms on a Grid, a testbed namely 'TUGrid Testbed' has been setup. Architectural layout of the TUGrid Testbed is shown in Figure 5.1. Hiding all the complexities from the user, this testbed provides an interactive, easy to use user interface. It uses open source software, technologies, standards and techniques in conjunction with existing Grid

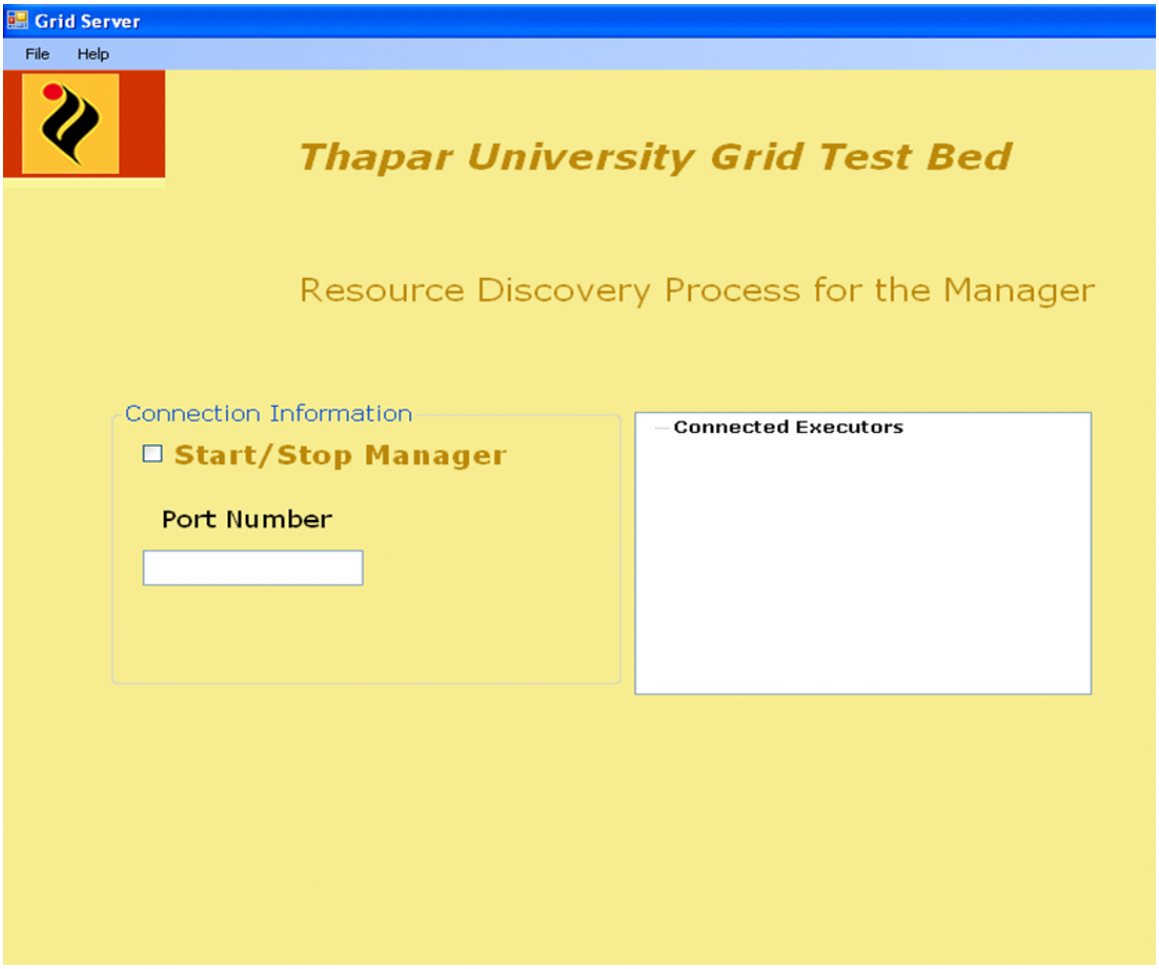



Figure 5.2: Resource Discovery Manager

infrastructures to facilitate access to Grid resources. Figure 5.5 shows the snapshot of live network of the University where all the experimentation has been performed. Figure 5.6 shows the Resource Discovery process in the campus Grid. The TUGrid Testbed has an architecture for integrating a number of existing technologies under a common interface. TUGrid Testbed consists of following components:

**Resource Pool:** TUGrid Testbed consists of clusters formed by different resources available in various departments, schools and hostels of Thapar University. These resources vary in their configuration, so as to form the complete heterogeneous

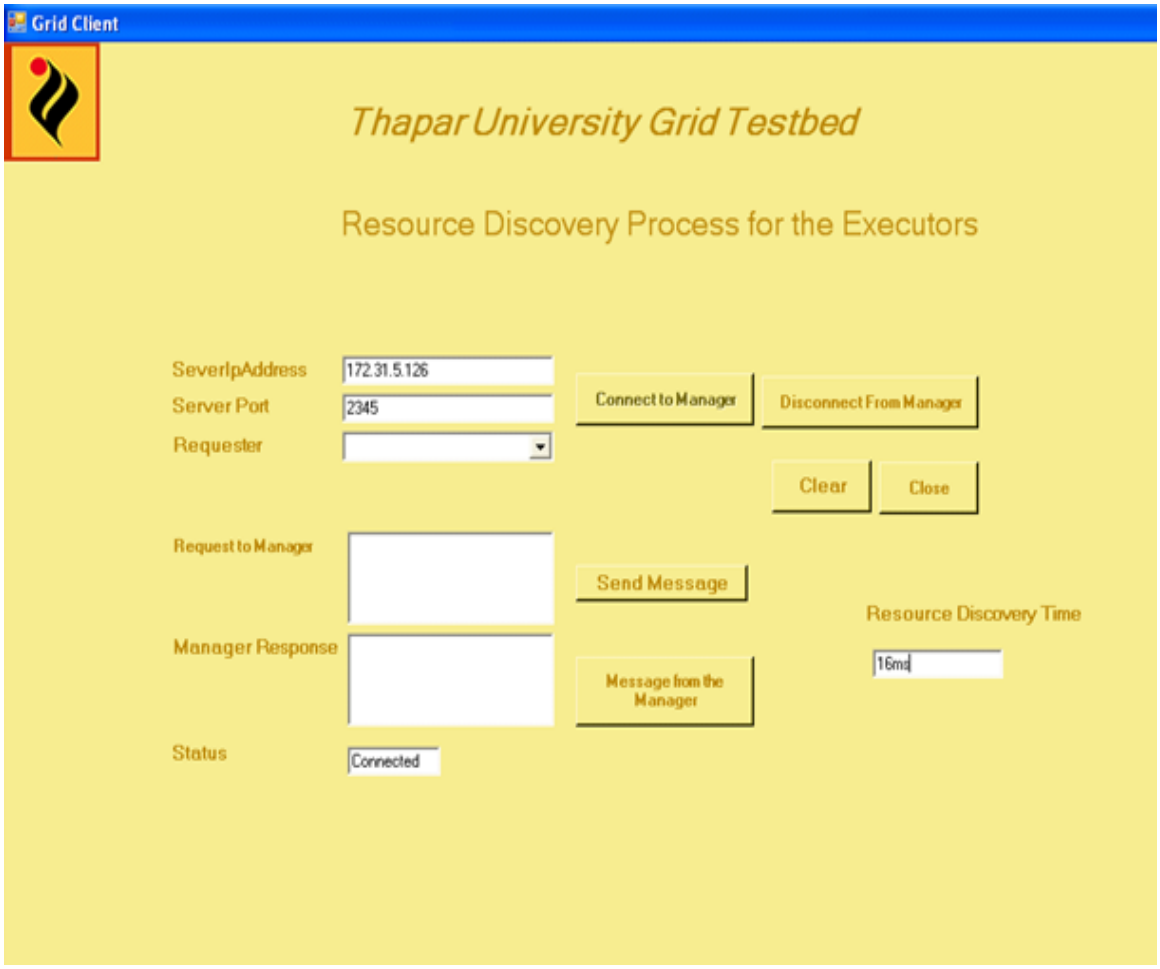



FIGURE 5.3: Resource Discovery Executor

environment. Each resource in each different clusters has its own security policy.

**Resource Discovery Manager:** Manager interface is shown in Figure 5.2. Its role is to manage the different resources registered into the resource pool. It keeps on monitoring about the current status of the resources.

**Resource Discovery Executor:** It performs the Resource Discovery process based on the all three proposed algorithms. A snapshot of an executor is shown in Figure

Table 5.1: Services defined as Exposed by the Manager and Executor

| Service Name | Purpose |
|---|---|
| WaitForData() | To take the request from the executor |
| AsyncCallback() | To data transfer intermittently |
| GetHostName() | To get the Particular Host Name |
| GetBytes() | To get the Data in bytes form |
| OnReceive() | On receiving the Data |
| AllocateResource() | To Allocate the Particular Resource |
| SetMessage() | To send the Messages Appropriately |

5.3. All these components are implemented as collection of different services. Related services are exposed as web services. Some of the services defined, exposed and used in the implementations are shown in Table 5.1:

Flow of information from the Manager to executor and different matching scenarios are shown in Figure 5.4. SQLserver database has been used to store all the results. During each run of the experimentation, the result is automatically stored in the corresponding tables. Through this test bed initially, the performance has been monitored inside within a department, *i.e.* CSED. Further performance is tested with other departments, schools and different hostels within the campus. In the testbed there is provision of different configurations of the resources for the requester. Following criterias are considered for the experimentation to prove the efficiency of the algorithms as given below:

- One client can request single resource or multiple resources, but more than

one client can not get the same resource allotted at the same time.

- All the different algorithms have different criteria to select the resources, *e.g.* minimum path based, minimum weight based, priority based, exact hit, multiple match *etc.*

- Evaluation of these algorithms has been made using the different matchmaking criteria as laid down in chapter 4.

The implemented algorithms are tested with the testbed on the Live Network having heterogeneous environment of Thapar University *i.e.* 'TUGrid Testbed' as shown in Figure 5.5. The experimentation is carried out for the different kinds of parameters including time, cost, reliability and security as explained in Chapter 3 of the thesis.

## 5.2 Evaluation Parameters

The implemented algorithms have been evaluated based upon the different performance metrics as explained in chapter 3 and chapter 4. Description of these metrics are as follows.

- Time: Time means the duration between the arrival of service request and the completion of the requested service.

- Cost: Cost represents the cost associated with the execution of the Grid tasks. Also explained in chapter 3 of thesis.

- Resource Matching: Process of matching the resources under different scenarios.

- Reliability: It defines the consistency and reliability of a Grid system to maintain its level of performance under stated condition for stated period of time.

- Security: It deals with preventing the Grid systems from any unauthorized access to maintain its confidentiality, integrity, reliability and availability of resources.

## 5.3 Experimental Results

Three algorithms have been deployed on the TUGrid testbed and the results have been reported in this chapter.

### 5.3.1 Test Cases

Different test cases are taken for the evaluation are as given below:

**Test Case 1 - *Within a Department (CSED)***

In this testcase Grid resources are confined to within a department. As a test case Computer Science and Engineering Department (CSED) has been considered. Table 5.2 shows the description of heterogeneous resources in the same department.

**Test Case 2 - *Across the different Departments in the Campus***

In this test case Resource Discovery algorithms are tested for the resources that are distributed across the various departments in the Thapar university. The Departments which are considered for the testbed are CSED, ECED, EIED, MED, BTSED and CED. Configuration of resources for the different departments are shown in Table 5.3.

**Test Case 3 - *Across the different Schools in the Campus***

In this test case Resource Discovery algorithms are tested for the resources that are distributed across the various Schools in the Thapar university. The Schools

Table 5.2: Snapshot of Resource Description Table with CSED

| ResourceID | Processor | Processor Speed(GHz) | RAM(MB) | HardDisk(GB) | Operating System |
|---|---|---|---|---|---|
| 1 | Pentium II | 2.8 | 2000 | 80 | Windows2000 |
| 2 | Pentium III | 1.5 | 512 | 20 | Windows2000 |
| 3 | Pentium 4 | 2.5 | 1024 | 80 | Fedora8 |
| 4 | Pentium III | 2.2 | 1024 | 60 | Solaris10 |
| 5 | Pentium 4 | 2.5 | 512 | 40 | WindowsXP |
| 6 | Pentium 4 | 2.3 | 1024 | 80 | Fedora8 |
| 7 | Pentium I | 2.5 | 1024 | 20 | RedHat9 |
| 8 ~ ~ ~ | Pentium 4 ~ ~ ~ | 2.5 ~ ~ ~ | 1024 ~ ~ ~ | 80 ~ ~ ~ | Fedora9 ~ ~ ~ |
| 24 | AMD | 2.5 | 1024 | 80 | Fedora9 |
| 25 | C2D | 3.5 | 2048 | 80 | Solaris10 |

which are considered for the testbed are SMCA, SCBC, SPMS, SMSS and LMTSM. Configuration of resources are shown in Table 5.4.

Table 5.3: Snapshot of Resource Description table with Different Departments

| ***ResourceID*** | ***Processor*** | ***Processor Speed(GHz)*** | ***RAM(MB)*** | ***HardDisk(GB)*** | ***Operating System*** |
|---|---|---|---|---|---|
| 1 | Pentium II | 3 | 1024 | 20 | WindowsXP |
| 2 | Pentium 4 | 2.5 | 256 | 60 | Linux |
| 3 | Pentium III | 4 | 1024 | 80 | Soalris10 |
| 4 | Pentium III | 4.5 | 512 | 20 | Windows98 |
| 5 | Pentium 4 | 2.6 | 512 | 40 | Windows2000 |
| 6 | Pentium II | 2.8 | 256 | 40 | WindowsXP |
| 7 | Pentium III | 3.1 | 256 | 20 | WindowsVista |
| ~ | ~ | ~ | ~ | ~ | ~ |
| ~ | ~ | ~ | ~ | ~ | ~ |
| ~ | ~ | ~ | ~ | ~ | ~ |
| ~ | ~ | ~ | ~ | ~ | ~ |
| 24 | Pentium III | 2.5 | 512 | 80 | WindowsXP |
| 25 | Pentium 4 | 3.0 | 512 | 60 | WindowsVista |

**Test Case 4 -** ***Across the different Hostels in the Campus***

In this test case Resource Discovery algorithms are tested for the resources that are distributed across the various Hostels in the Thapar university. The Hostels which are considered for the testbed are A, B, C, D, E, I and G. Configuration of resources are shown in Table 5.5.

Table 5.4: Snapshot of the Resource Description table with different Schools in the Campus

| ***ResourceID*** | ***Processor*** | ***Processor Speed(GHz)*** | ***RAM(MB*** | ***HardDisk(GB)*** | ***Operating System*** |
|---|---|---|---|---|---|
| 1 | Pentium II | 2.2 | 512 | 10 | Windows98 |
| 2 | Pentium 4 | 4 | 512 | 160 | Linux |
| 3 | Pentium III | 3.2 | 512 | 40 | Windows2000 |
| 4 | C2D | 2.0 | 1024 | 320 | Solaris9 |
| 5 | Pentium III | 2.3 | 256 | 40 | Windows2000 |
| 6 | Pentium II | 3.2 | 256 | 20 | Windows98 |
| 7 | Pentium II | 2.9 | 256 | 80 | WindowsXP |
| ∼ ∼ ∼ ∼ | ∼ ∼ ∼ ∼ | ∼ ∼ ∼ ∼ | ∼ ∼ ∼ ∼ | ∼ ∼ ∼ ∼ | ∼ ∼ ∼ ∼ |
| 24 | AMD | 2.5 | 512 | 80 | WindowsXP |
| 25 | Pentium II | 2.5 | 512 | 60 | WindowsVista |

### 5.3.2 Testing Results

This section presents the performance, "Path Optimization", "Weight Optimization" and "Request Matching" Algorithms. In Path Optimization algorithm the parameter: time taken have been analyzed practically on heterogeneous resources using different plots *i.e.* time *vs.* within department (CSED), time *vs.* with different departments, time *vs.* with different schools and time *vs.* with department hostels as depicted in Figures 5.7, 5.9, 5.11 and 5.13. Time taken by this algorithm in CSED, different departments, different schools and different hostels are tabulated

Table 5.5: Snapshot of Resource Description table with different Hostels in the campus

| ***ResourceID*** | ***Processor*** | ***Processor Speed(GHz)*** | ***RAM (MB)*** | ***HardDisk(GB)*** | ***Operating System*** |
|---|---|---|---|---|---|
| 1 | Pentium II | 4 | 1024 | 20 | Windows98 |
| 2 | Pentium 4 | 2.4 | 256 | 20 | Linux |
| 3 | Pentium 4 | 3.1 | 512 | 160 | Linux |
| 4 | Pentium II | 2.9 | 256 | 60 | Fedora9 |
| 5 | Pentium 4 | 4.2 | 1024 | 80 | Solaris9 |
| 6 | Pentium III | 3.2 | 256 | 40 | Windows2000 |
| 7 | Pentium III | 1.9 | 128 | 40 | Windows98 |
| 8 | Pentium II | 1.8 | 256 | 40 | Windows98 |
| ~ ~ ~ | ~ ~ ~ | ~ ~ ~ | ~ ~ ~ | ~ ~ ~ | ~ ~ ~ |
| 24 | Pentium 3 | 2.5 | 512 | 80 | WindowsXP |
| 25 | Pentium 4 | 2.5 | 1024 | 60 | WindowsVista |

in Tables 5.6, 5.7, 5.8 and 5.9. These results are highly encouraging and demonstrate the effectiveness of the proposed algorithm and it will ultimately result into discover the efficient resource in reasonable amount of time by optimally utilizing the heterogeneous resources available in TUGrid Testbed.

Table 5.6: Time comparison of Resource Discovery process across CSED

| *Computer Department* | *CoEGC* | *CCCT* | *SE* | *CCMX Lab* | *CSMC* |
|---|---|---|---|---|---|
| *Time Taken in Milliseconds (ms)* | 16 | 25 | 22 | 28 | 18 |

The Weight Optimization Algorithm is designed in such a way that it keeps costs as the highest priority. Testing of the same has been done using the following scenarios:

A) A higher speed processor is available at a location which is far away from the central point. The proposed algorithm successfully selects the given processor.
B) In this scenario higher speed processor was within one hop away from the central point. The algorithm is successfully able to discover the same. And hence, it optimizes the cost that is vary within department, department to department, school to school, and hostel to hostel *etc.*

In Weight Optimization algorithm the parameter: costs of the resources have been analyzed practically on heterogeneous resources using different plots *i.e.* cost per unit time *vs.* within department *i.e.* CSED, cost per unit time *vs.* with different departments, cost per unit time *vs.* with different schools, cost per unit time *vs.* with different hostels and time *vs.* with CSED, time *vs.* with different departments, different schools and different hostels and number of hops *vs.* with CSED, number of hops *vs.* with different departments, number of hops *vs.* with different schools ,number of hops *vs.* with different hostels are depicted in Figures 5.8, 5.10, 5.12 and 5.16. These results are highly encouraging and demonstrate the effectiveness of the proposed algorithm and it will ultimately result into discover the efficient resource

Table 5.7: Time comparison of Resource Discovery process across different Departments

| *Different Departments* | *MED* | *BTESD* | *ECED* | *CSED* | *EIED* | *DDE* | *CED* |
|---|---|---|---|---|---|---|---|
| *Time Taken in Milliseconds (ms)* | 92 | 83 | 42 | 19 | 29 | 90 | 72 |

in reasonable amount by optimally utilizing the heterogeneous resources available in TUGrid Testbed.

In the concluding remarks of both these algorithms, when the resources are more then six hops away the time taken in discovery process is more. As can be seen from the Table 5.9 Resource Discovery process for the resources located in the hostels shows significantly much time taken as compare to different departments and schools. Hostels A, B, C, H, and I are more then six hops away from the central point. Also the latency can be attributed due to the uncontrolled traffic flowing from the hostels like torrents traffic for big downloads or compromise machines.

The Request Matching Algorithm helps to find the best resource from a set of all matching resources for a given job, as it works with the different matching scenarios and three matchmaking principles as discussed in previous Chapter. Matches of the resources have been analyzed practically and results obtained from them are shown in Figures 5.15, 5.16, 5.17 and 5.18. Where as graph plotting the time *vs.* number of resources is depicting in Figure 5.20. These results are highly encouraging and

Table 5.8: Time comparison of Resource Discovery process across different Schools

| *Different Schools* | *LMTSM* | *SCBC* | *SMSS* | *SMCA* | *SPMS* |
|---|---|---|---|---|---|
| *Time Taken in Milliseconds (ms)* | 183 | 84 | 73 | 32 | 94 |

demonstrate the effectiveness of the proposed algorithm and it will ultimately result into discover the efficient resource in reasonable amount and with optimal time.

Reliability: TUGrid exhibits the reliability by means of data duplication simultaneously to the remote server. In case of any failure database can be makeup by running scripts from the remote location. Also administrator can schedule a checksum on the database tables to verify the size on the master as well as backed up database.

Security: As can be seen from the Figure 5.19 only authorized requester get a chance to authenticate their credentials against the security parameter. When a requester wishes to get services from TUGrid it is first mapped as a requester into database followed by authentication processing. Authentication is implemented using encrypted signatures stored in the database. TUGrid uses MD5 encryption for the said purpose.

Table 5.9: Time comparison of Resource Discovery process across different Hostels

| *Different Hostels* | *Hostel A* | *Hostel B* | *Hostel C* | *Hostel H* | *Hostel I* |
|---|---|---|---|---|---|
| *Time Taken in Milliseconds (ms)* | 297 | 252 | 202 | 299 | 117 |

## 5.4 Conclusions

This chapter establishes the effectiveness of the Resource Discovery algorithms by fulfilling the Resource Discovery requirements optimally in the campus Grid. In this chapter all the algorithms have been deployed, tested and validated on the 'TUGrid Testbed' and the experiment shows the time, cost, matching scenarios, reliability and security. To achieve the reliability Resource Rediscovery Manager keeps on working fine even when one of the node stop working in-between. Our resource discovery approach helps to modify the resource list dynamically with respect to the current status of resources participating into the grid environment. Another important factor that has been taken care is Security of grid environment. Proper authentication of users has been done, before assigning any resources to its submitted job as shown in Figure 5.19.

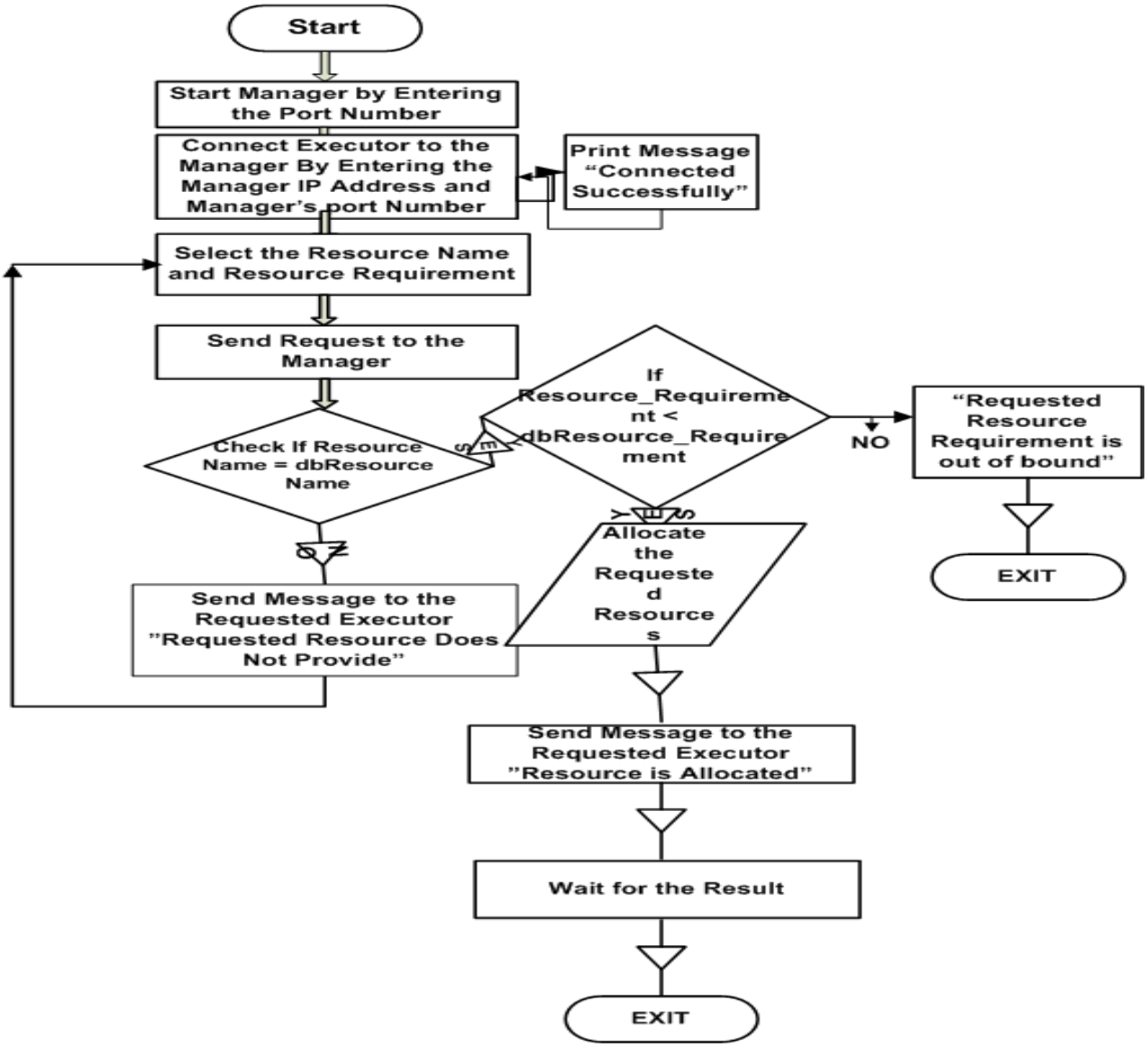



Figure 5.4: Flow of information in the Testbed

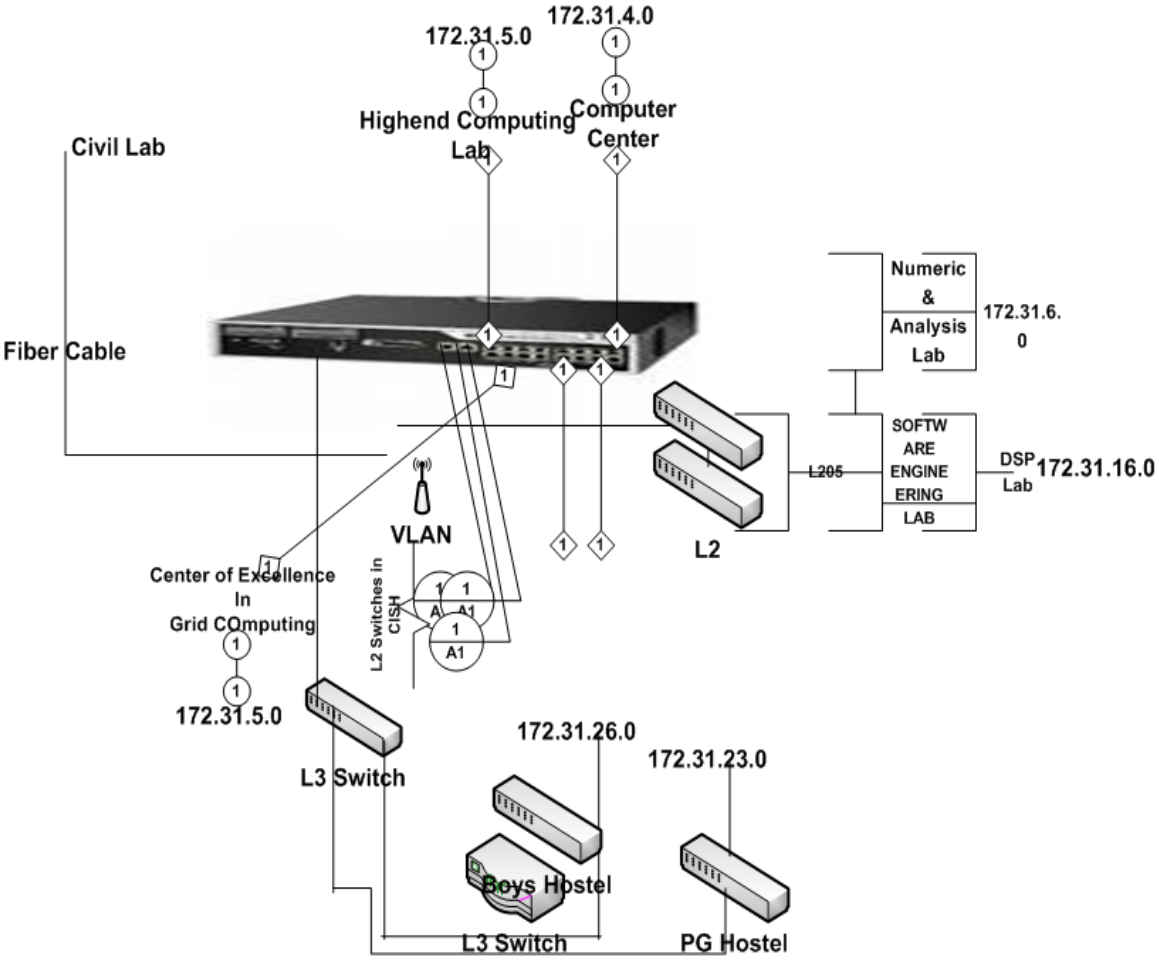



Figure 5.5: Testing with Live network at Thapar University

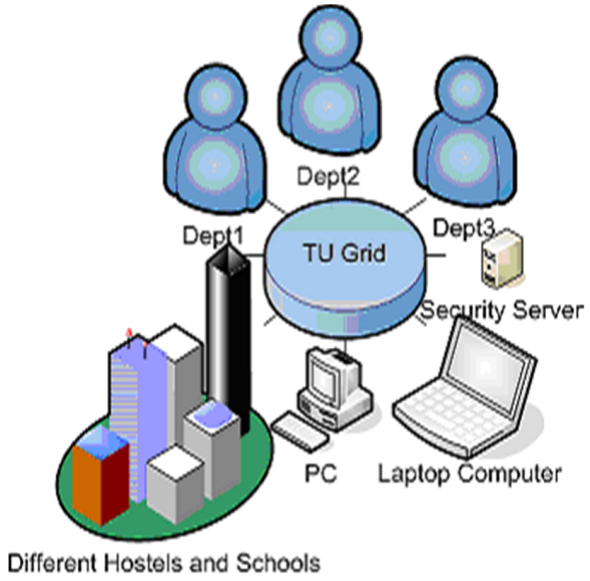



Figure 5.6: Resource Discovery Process in Campus Grid

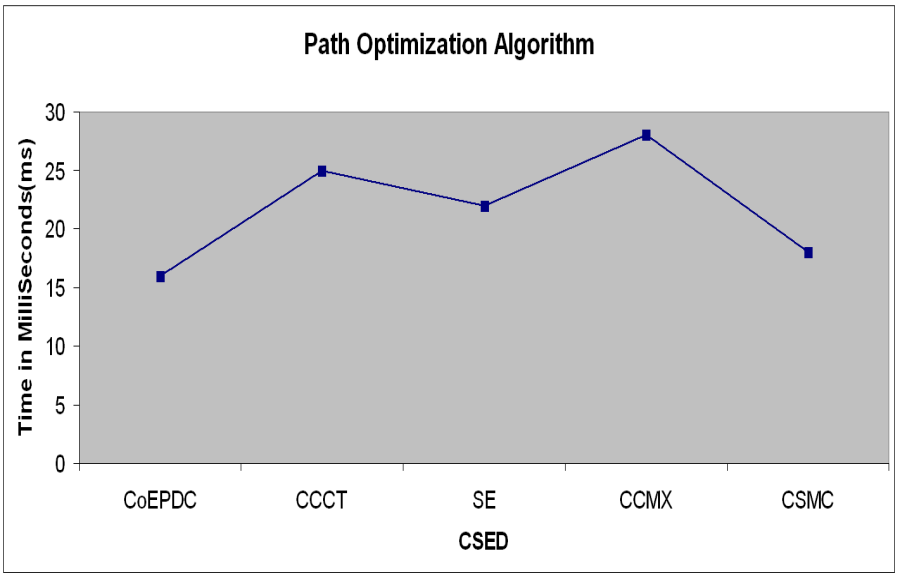



Figure 5.7: Graph depicting time taken with CSED in Path Optimization Algorithm

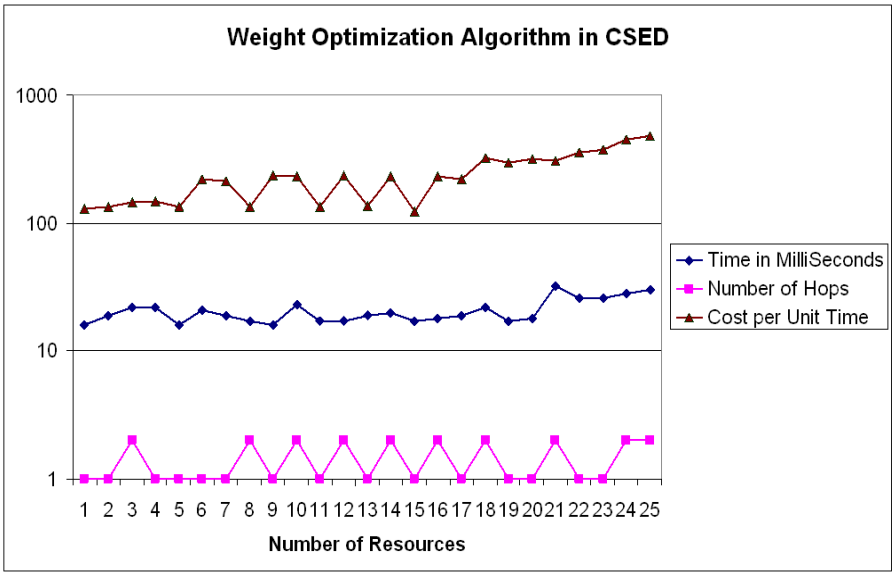



Figure 5.8: Graph depicting cost with CSED in Weight Optimization Algorithm

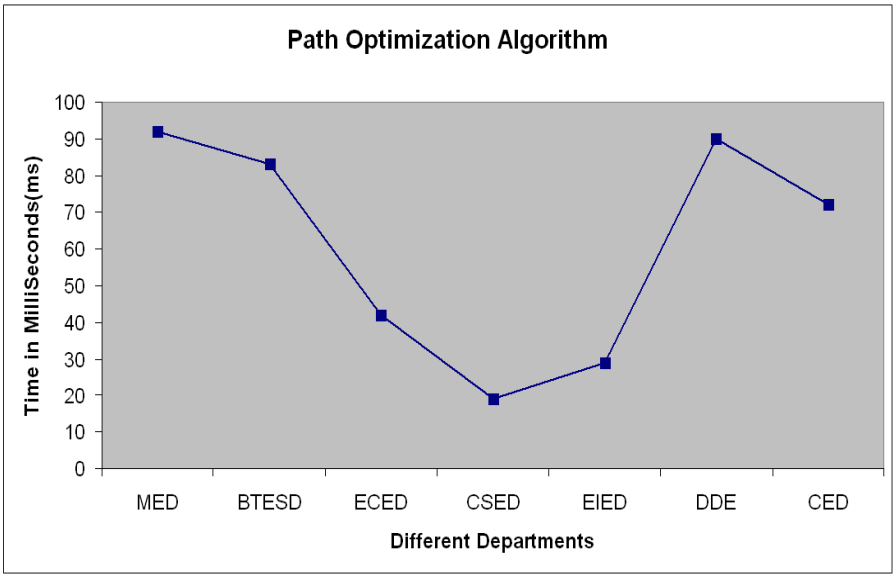



Figure 5.9: Graph depicting time taken with different Departments in Path Optimization Algorithm

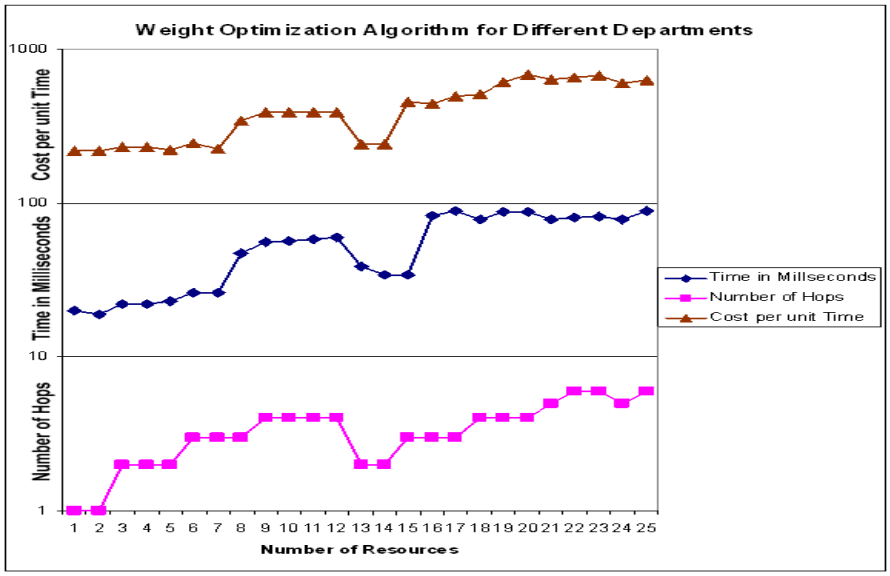



FIGURE 5.10: Graph depicting cost with different Departments in weight Optimization Algorithm

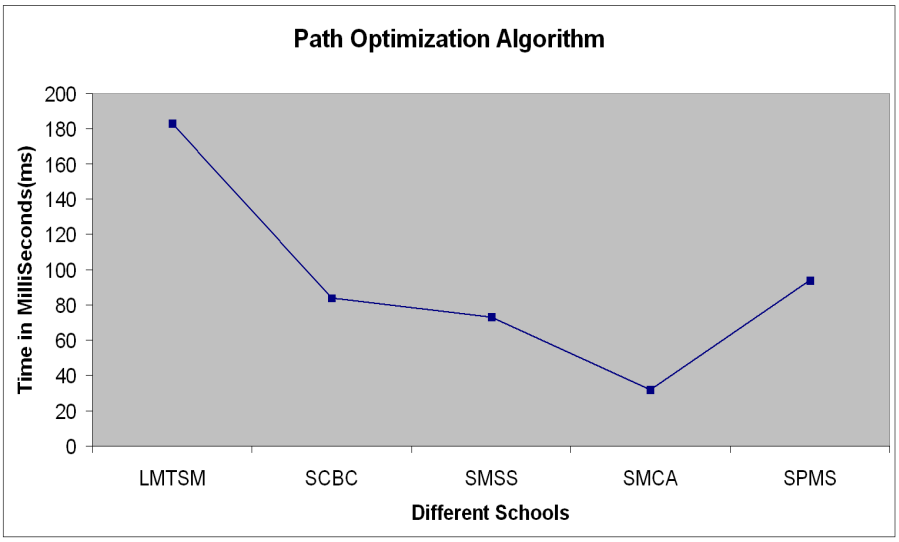



FIGURE 5.11: Graph depicting time taken with different Schools

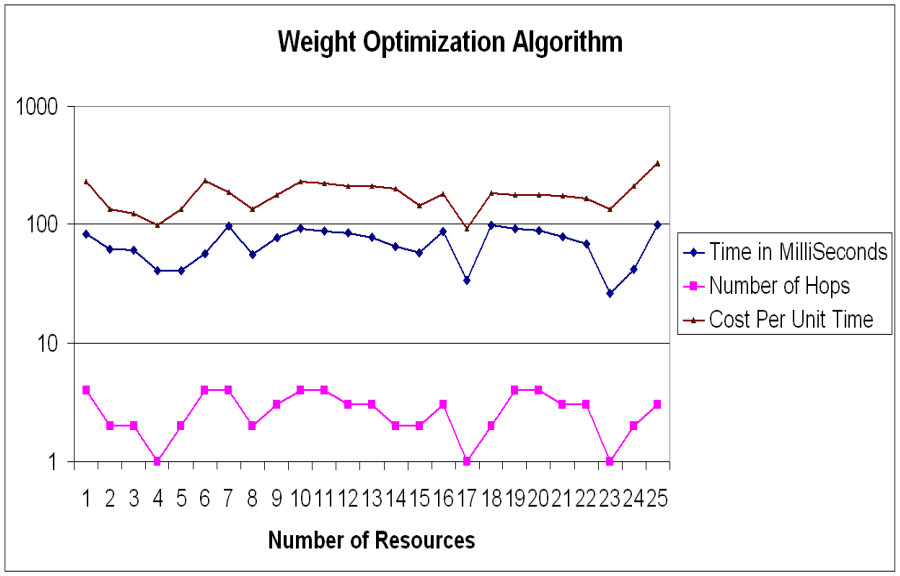



Figure 5.12: Graph depicting cost with different Schools in weight Optimization Algorithm

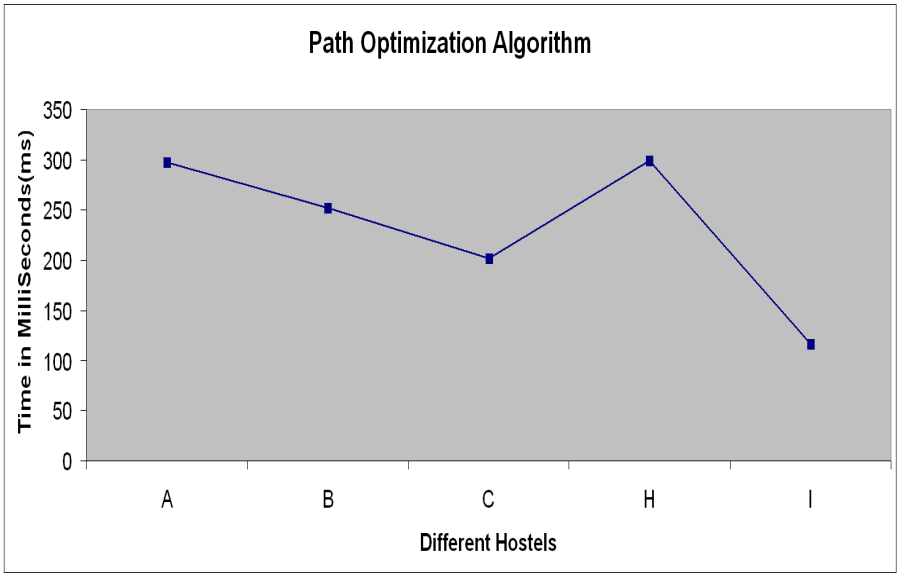



Figure 5.13: Graph depicting time taken in different Hostels in Path Optimization Algorithm

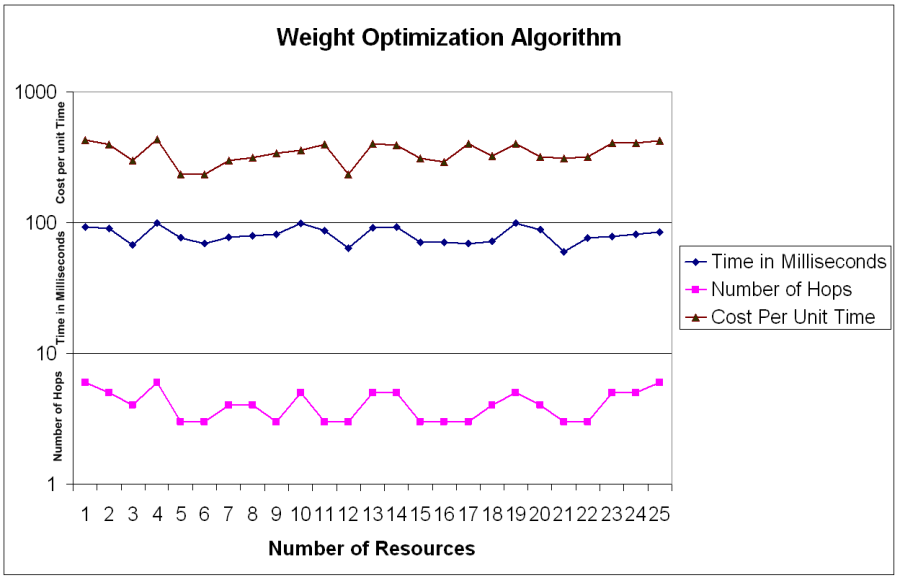
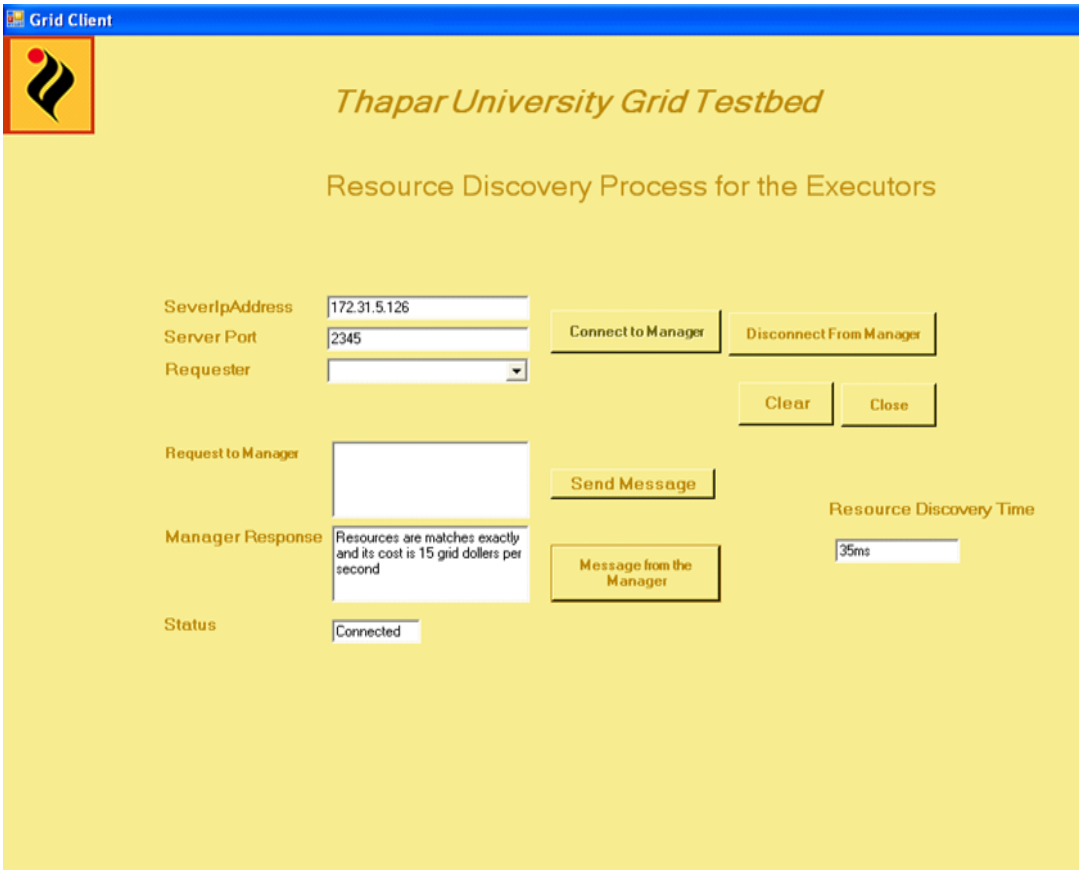



Figure 5.14: Graph depicting cost in different Hostels in Weight Optimization Algorithm



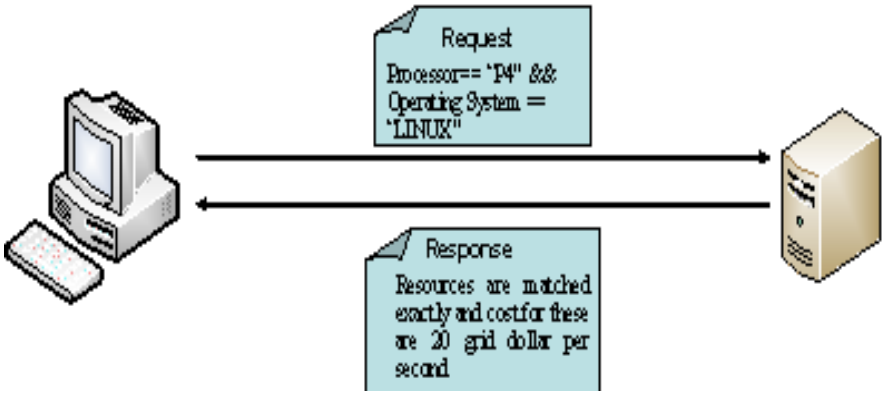
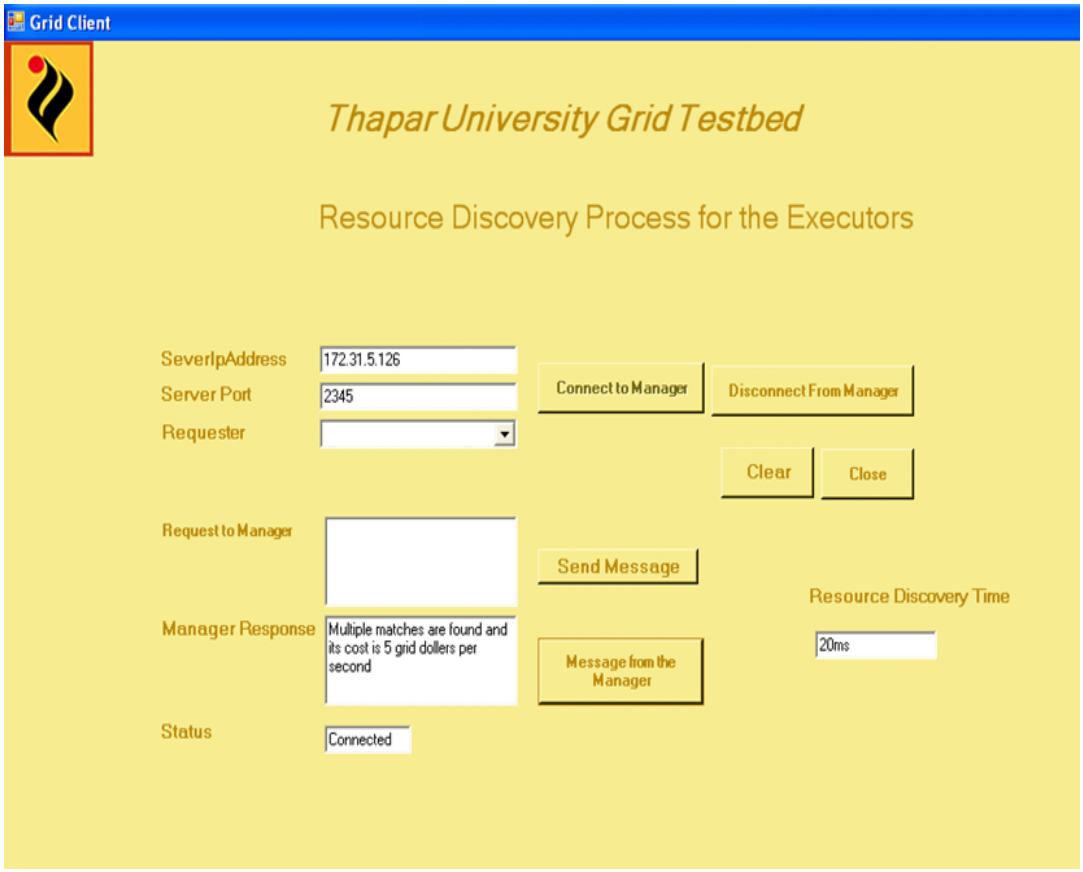




Figure 5.15: Exact Hit Scenario



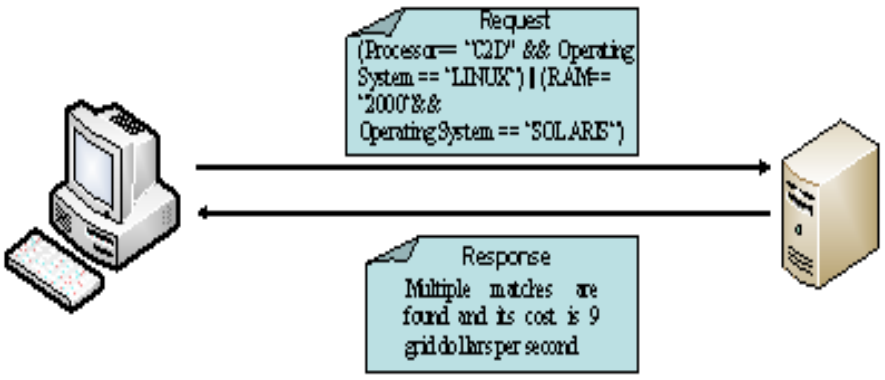
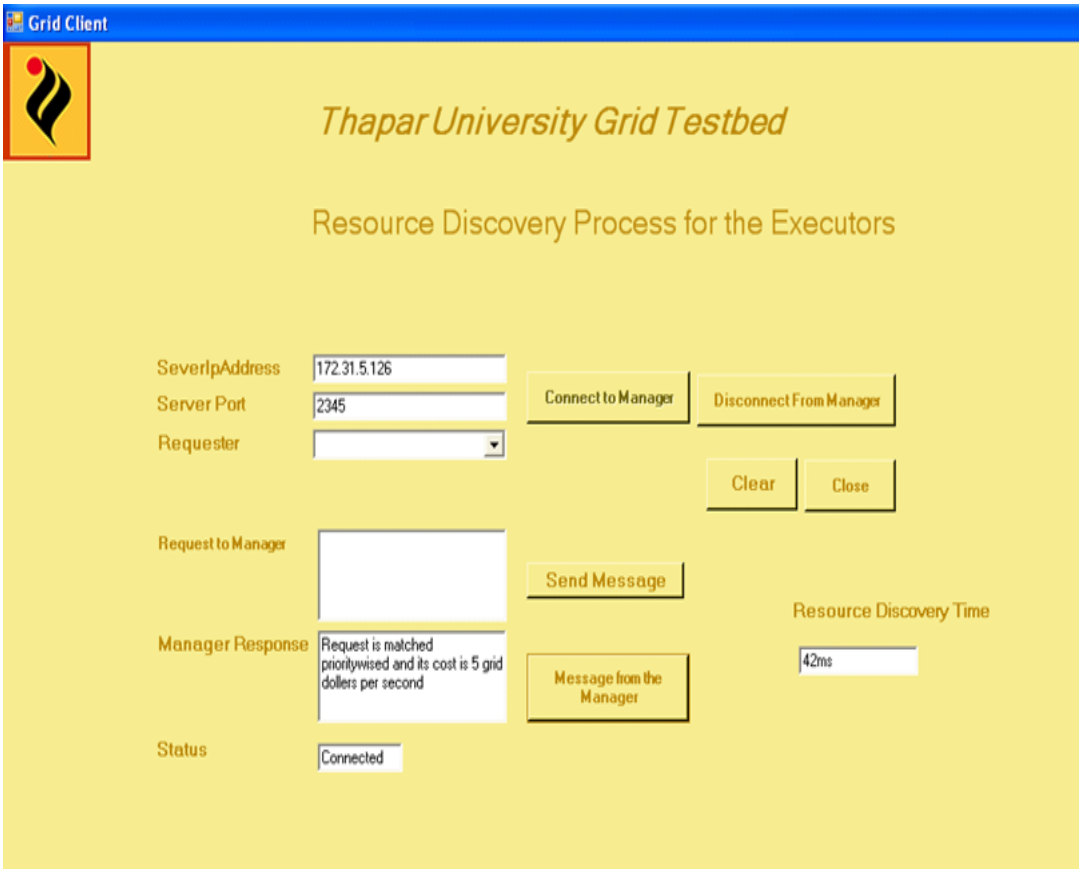




Figure 5.16: Multiple Match Scenario



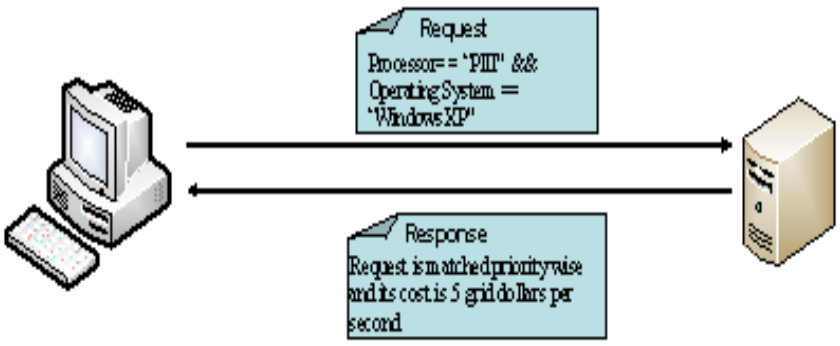




Figure 5.17: Priority based Match Scenario

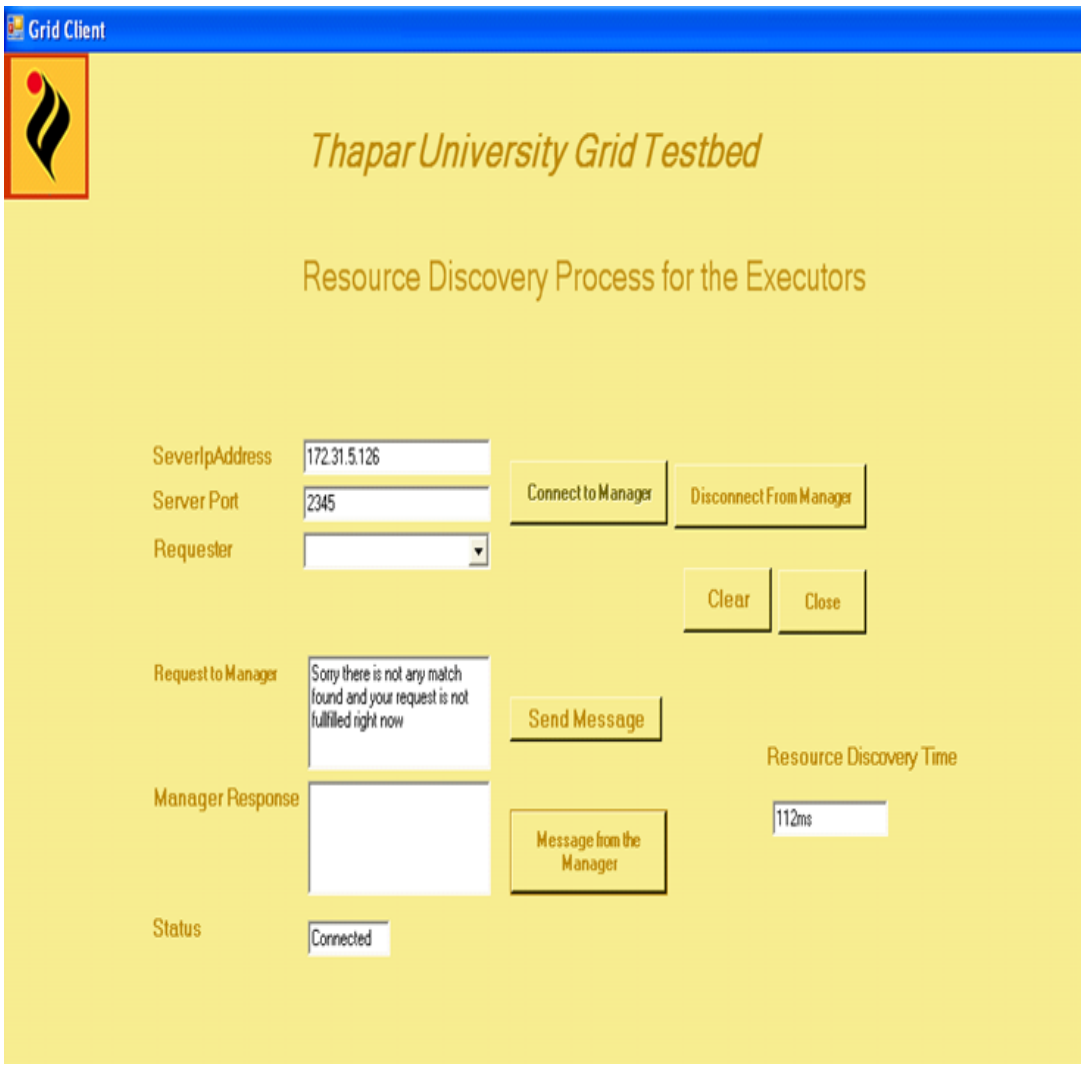




Figure 5.18: Miss Scenario

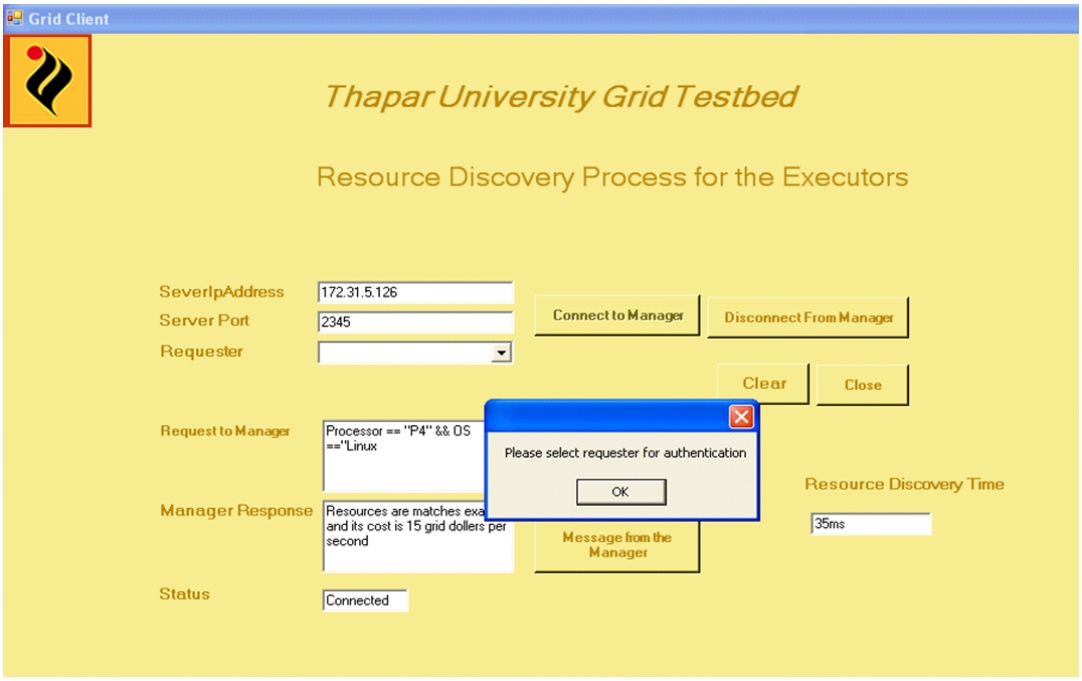



Figure 5.19: Snapshot of the Security Parameter

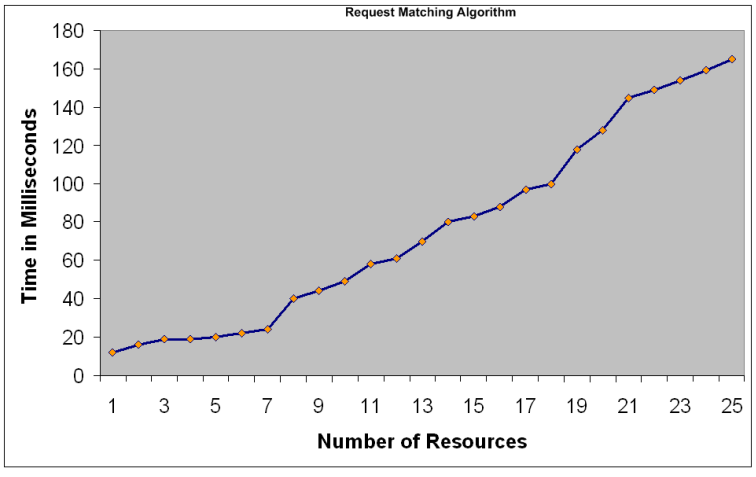



Figure 5.20: Graph showing Time taken by Request Matching Algorithm

# 6

# Conclusions and Future Scope of the Work

*In Grid environments, the resources are heterogeneous and geographically distributed with varying availability and a variety of usage and cost policies for diverse users at different times and, priorities as well as goals that vary with time. The discovery of resources in such a large and distributed environment is a complex task. The problem in such systems is how Resource Discovery should be done in the case where some resources are lying idle and could be linked with other overloaded nodes in the network. In this thesis, the issues and challenges involved in efficient Resource Discovery in heterogeneous Grid environment has been identified. To overcome such issues this thesis proposes three different algorithms for Resource Discovery in dynamic Grids*

*and evaluate the efficiency. This chapter provides some concluding remarks, highlights the main contribution of this thesis and presents future research directions. Organization of the chapter is as follows: Section 6.1 concludes the research work carried out in this thesis. Section 6.2 presents the main contributions of this thesis. Section 6.3 highlights the several possible future research directions.*

## 6.1 Conclusions

Grid Computing is an emerging form of distributed computing, distinguished from traditional forms by its focus on large-scale, multi-organizational resource sharing and innovative applications. Like any computing infrastructure, the Grid infrastructure is made up of hardware and software, which have been advancing rapidly. This advance is facilitated greatly by innovations in building fast and powerful commodity computers and networks. To summarize the work presented in this thesis, provides an insight into the world of Grid Computing and the current state of the art in Grid Resource Discovery Algorithms along with pros and cons of each approach and algorithms. This thesis entitled *Efficient Resource Discovery in Grid Environments* can deliver significant value to researchers working in the area of Grid Computing.

Grid Computing, supports the varied resources, and it needs a suite of Resource Discovery algorithms. It has been realized that, in order to carry out the work with resource discovery, it is useful to develop some approaches, models and algorithms that support the heterogeneous nature of resources. Proposed algorithms namely: *Path Optimization Algorithm*, *Weight Optimization Algorithm*, and *Request Matching Algorithm*, provide support to take care of various different QoS parameters, in accurate amount of time and with optimal cost. The implementations carried out and results obtained so far demonstrate that the identified objectives have been achieved and the approach is workable.

## 6.2 Thesis Contributions

- In depth description of Resource Discovery process and related issues has been made.
- Initially, in depth review of existing Resource Discovery models, approaches and algorithms has been done.
- In Literature Review, a comprehensive analysis of different approaches has been laid down. Also a comparative analysis of different Middleware has been done.
- Detailed study of the QoS aware Resource Discovery process in Grid computing environment has been made, different QoS parameters have been analyzed (including Time, Cost, Security and Reliability).
- After considering the comparative analysis of all the aspects of Resource Discovery three algorithms namely: "Path Optimization Algorithm", "Weight Optimization Algorithm" and "Request Matching Algorithm" have been proposed, designed and developed using the different approaches, scenarios and assumptions.
- The testbed namely "TUGrid tesbed" has been designed and implemented to test the resource discovery algorithms.
- All the three algorithms: *Path Optimization Algorithm*, *Weight Optimization Algorithm*, and *Request Matching Algorithm*, have been deployed and tested on the testbed. The performance of the algorithms has been successfully demonstrated by taking the different QoS parameters under considerations in different testcases.
- The Validation of the algorithms has been done with the TU Grid testbed, results demonstrate the efficiency of the algorithms.

- After validation, comprehensive comparison of all the three algorithms has been made with existing Resource Discovery algorithms.

## 6.3 Future Scope of the work

To meet new challenges in Resource Discovery process in a heterogeneous Grid environment, different QoS parameters like (Time, Cost, Reliability and Security) and three Resource Discovery algorithms have been proposed in this thesis namely "Path Optimization", "Weight Optimization" and "Request Matching Algorithm". This work can be enhanced in future in the following ways:

- More QoS parameters like throughput, latency, ability *etc.* can be considered for implementation of above algorithms .
- Weight Optimization Algorithm can further be enhanced by considering best resource configuration match.
- Inherently parallel algorithms can be developed and designed to get more efficient Resource Discovery
- Request Matching Algorithm can be extended to include more kinds of scenarios and principles like partial matching, bipartite matching can be considered.
- Tools like Planet Lab can be used for enhancing the reliability of algorithms by testing them more efficiently

# Bibliography

[AA04] Atif Mehmood Richard McClatchey Ian Willers-Julian Bunn Harvey Newman Michael Thomas Conrad Steenberg Arshad Ali, Ashiq Anjum. *A Taxonomy and Survey of Grid Resource Planning and Reservation Systems for Grid Enabled Analysis Environment*. International Symposium on Distributed Computing and Applications to Business Engineering and Science, ACM, 2004.

[Abb04] Ahmar Abbas. *Grid Computing: A practical guide to technology and Applications, ISBN:81-7008-626-4*. Firewall Media, 2004.

[AK83] S. Assmann and D. Kleitman. *The number of rounds needed to exchange information within a graph*. SIAM Discrete Applied Math, pp 117-125, 1983.

[ALV05] R. Ranjan A. Luther, R. Buyya and S. Venugopal. *High Performance Computing: Paradigm and Infrastructure, New Jersey, USA*. Wiley Press, 2005.

[AP05] M.A.R Dantas A.M. Pernas. *Using Ontology for Description of Grid Resources*. 19th IEEE International Symposium on High Performance Computing Systems and Applications, 2005.

[AVA03] Jefferey D. Ullman Alfred V. Aho, John E. Hopcroft. *The Design and Analysis of Algorithms*. Pearson Education, ISBN: 81-7808-103-2, 2003.

[AVA04] Jefferey D. Ullman Alfred V. Aho, John E. Hopcroft. *Data Structures and Algorithms*. Pearson Education, ISBN: 81-7808-102-4, 2004.

[Baw04] Seema Bawa. *Parallel and Distributed Algorithms for Test Generation*. PhD thesis, Thapar Deemed University, Patiala, 2004.

[BE07] Jeremy Buisson and Dick Epema. *Grid Resource Allocation and Reallocation for Adaptable Applications*. Report of CoreGRID, REP 9 (CR14 INRIA - CR06 TUD), May 2007.

[BJ03] Norbert Bieberstein Candice Gilzean Jean-Yves Girard Roman Strachowski Seong (Steve) Yu Bart Jacob, Luis Ferreira. *Enabling Applications for Grid Computing with Globus*. IBM.com/redbooks, International Technical Support Organization, 2003.

[BTY02] Mike Duigou Jean-Christophe Hugly-Eric Pouyoul Bernard Traversat, Mohamed Abdelaziz and Bill Yeager. Project jxta virtual network, 2002.

[CL00] Kai-Yeung Siu C. Law. *An O(log n)randomized resource discovery algorithm*. 14th International Symposium on Distributed Computing Technical Report, Technical University of Madrid, pp 5-8, 2000.

[Cor08] Jupitermedia Corporation. *http://www.gridcomputingplanet.com/faq/#two*. Grid Computing Planet.com, 2008.

[CS97] J.P. LeBlanc Chris Sluman, Jeremy Tucker. *Quality of Service*. OMG Green Papaer, OMA/SORBA, 1997.

[DAM04] Emiliano Casalicchio Daniel A. Menasc. *QoS in Grid Computing.* Internet Computing, IEEE, Volume 8, Issue 4, ISSN: 1089-7801, pp 85-87, 2004.

[DAS97] A. Abbadi D. Agrawal and R. Steinke. *Epidemic Algorithms in Replicated Databases.* 16th Symposium on Principles of Database Systems, ACM SIGACT-SIGMOD-SIGART, pp 161-172, 1997.

[DB05a] Habiba DRIAS Dalila BOUGHACI. *An Agent-based Approach using the ebXML Specifications for E-business.* International Conference on Information Technology: Coding and Computing, 2005.

[DB05b] Habiba DRIAS Dalila BOUGHACI. *Taboo search as an intelligent agent for bid evaluation.* International Journal of Internet and Enterprise Management, Volume 3, Issue 2, pp 170-186, 2005.

[DC06] Y. Hassoun et al. D. Colling, T. Ferrari. *On Quality of Service Support for Grid Computing.* 2nd International Workshop on Distributed Cooperative Laboratories, Italy, 2006.

[Deo02] Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, India, 2002.

[DKT07] Theodora Varvarigou Fabrizio Silvestri Domenico Laforenza Dimosthenis Kyriazis, Andreas Menychtas and Konstantinos Tserpes. *An Open Architecture for QoS Information in Business Grids, ISBN: 978-0-387-72497-3 (Print) 978-0-387-72498-0 (Online).* Springer US, 2007.

[Dob04] Glen Dobson. *Quality of Service in SOA.* http://digs.sourceforge.net/papers/qos.html, Technical report, 2004.

[DTGC01] Claudio Bartolini David Trastour and Javier Gonzalez-Castillo. *A semantic web approach to service description for matchmaking of services.* International Semantic Web Working Symposium, Stanford, CA, USA, 2001.

[DTL05] Todd Tannenbaum Douglas Thain and Miron Livny. *Distributed Computing in Practice: The Condor Experience.* Concurrency and Computation: Practice and Experience, Volume 17, Issue 2-4, pp 323-356, 2005. http://www.cs.wisc.edu/condor/publications.html.

[Dur04] Jean-Christophe Durand. Grid computing: A conceptual and practical study. Master's thesis, University of Lausanne, 2004.

[ea05] Deelman E. et al. *Pegasus: A framework for mapping complex scientific workflows onto distributed systems.* Scientific Programming Journal, Volume 13, Issue 3, pp 219-237, 2005.

[ea06a] Babu Sundaram et al. *Sun Grid Engine Package for OSCAR.* 20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment, 2006.

[ea06b] Paolo et al. *Peer-to-Peer Models for Resource Discovery on Grids.* CoreGrid Technical Report Number TR-0028, 2006.

[EM05] Joan Serrat Edgar Magana. *QoS Aware Policy Based Management Architecture for Service Grids.* 14th IEEE International Workshop on Enabling Technologies: Infrastructure for Collborative Enterprise, 2005.

[Eng90] Sun Grid Engine. *Open Source Software, http://gridengine.sunsource.net.* Sun Grid Engine, 1990.

[FBW02] M. van Steen F.M.T. Brazier, B.J. Overeinder and N.J.E. Wijngaards. *Agent Factory: Generative Migration of mobile agents in heterogeneous environment.* In Proceedings of the AIMS Workshop, 2002.

[FG07] Behrouz A. Forouzan and Richard F. Gilberg. *Computer Science A Structured Programming Approach using C.* Thomson, Third Edition, ISBN: 81-315-0363-1, 2007.

[FK97] I. Foster and C. Kesselman. *Globus: A Metacomputing Infrastructure Toolkit.* International Journal of Supercomputer Applications, 1997. Volume 11, Issue 2, pp 115-128.

[FK05] Ian Foster and Carl Kesseleman. *The GRID2 blueprint for a New Computing Infrastrucure.* Morgan Kaufmann, Second Edition, ISBN: 1-55860-933-4, 2005.

[GC01] T. Kindberg G. Coulouris, J. Dollimore. *Distributed Systems: Concepts and Design.* Addison-Wesley, 2001.

[GCK01] Jean Dollimore George Coulouris and Tim KindBerg. *Distributed Systems Concepts and Design, Third Edition.* Pearson Education Asia, 2001.

[Gen01] Wolfgang Gentzsch. *Sun Grid Engine: Towards Creating a Compute Power Grid.* 1st International Symposium on Cluster Computing and the Grid, IEEE, pp 35-40, 2001.

[GKS02] Seema Bawa G K Sharma. *A Parallel Transitive Closure Computation Algorithm for VLSI Test Generation.* LNCS 2367, Springer Verlag, Berlin- Heidelberg, pp 243-252, 2002.

[Gru93] T. Gruber. *A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition.* Available [Online] http://www.ksl.stanford.edu/kst/what - is-an-ontology.html., 1993.

[Hos02] W. Hoschek. *A Unified Peer-to-Peer Database Framework for Scalable Service and Resource Discovery.* International Workshop on Grid Computing: GRID 2002, IEEE/ACM, Baltimore, MD., Springer, pp 126-144, 2002.

[HS95] Jason Sodergen Harpreet Singh. *Distributed Processing Techniques for Generation of Dertouzos Tables.* International Conference on Parellel and Distributed Computing Systems, Orlando, Florida, pp 337-342, 1995.

[HZS05] Zongfen Han Jing Tie Hongbo Zou, Hai Jin and Xuanhua Shi. *A Virtual-Service-Domain based Bidding Algorithm for Resource Discovery inComputational Grid.* International Conference on Web Intelligence ISSN: 0-7695-2415, IEEE/ACM, 2005.

[IA06] D. Dolev I. Abraham. *Asynchronous resource discovery.* International Journal of Computer and Telecommunications Networking, Volume 50, Issue 10, ISSN: 1389-1286, pp 1616-1629, 2006.

[Iam02] Daniel C. Nurmi Iamnitchi, I. Foster. *A Peer-to-Peer Approach to Resource Discovery in Grid Environments.* 11th Symposium on High Performance Distributed Computing, Edinburgh, UK, 2002.

[Iam03] Adriana Ioana Iamnitchi. *Resource Discovery in Large Resource sharing environment.* PhD thesis, Chicago, Illinois, 2003.

[ICJ02] Ho-Sang Ham In-Cheol Jeong, Tae-Gun Kang. *Serving Reliable Resource Discovery in Grid Environment.* ETRI, APNOM, 2002.

[IF99] Craig Lee Bob Lindell Klara Nahrstedt Alain Roy Ian Foster, Carl Kesselman. *A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation*. Proceedings of the International Workshop on Quality of Service, 1999.

[IF01a] S. Tuecke I. Foster, C. Kesselman. *The Anatomy of Grid: Enabling Scalable Virtual Organizations*. International Journal of Supercomputer Applications, 2001.

[IF01b] Iamnitchi and I. Foster. *On Fully Decentralized Resource Discovery in Grid Environments*. International Workshop on Grid Computing, IEEE, 2001. Denver, Colorado, USA, pp 51-62.

[IFT02] Jeffrey M. Nick Ian Foster, Carl Kesselman and Steven Tuecke. *The physiology of the Grid, Third Edition*. Global Grid Forum, 2002.

[IFT03] J. Nick I. Foster, C. Kesselman and S. Tuecke. *The Physiology of the Grid, New Jersey, USA*. Wiley Press, pp 217-249, May 2003.

[JA07] Y. Wu J. Aspnes. *O(logn)-time overlay network construction from graphs with out-degree 1*. LNCS, OPODIS, 2007.

[JF04] Joshy Joseph and Craig Fellenstein. *Grid Computing*. Prentice Hall/IBM Press, ISBN: 81-297-0527-3, 2004.

[JP03] San Diego Jim Pinto. *Distributed & Grid Computing*. Automation.com, http://www.jimpinto.com/writings/grid.html, May 2003. USA.

[Jul04] Wilfried Juling. *Condor Clusters with Multilateral Resource Matchmaking in Heterogenous Networks*. Diploma Thesis at the Institute of Telematics, 2004.

[JXT02] Project JXTA. Jxta v1.0 protocols specification, 2002.

[Kan03] Chaitanya Kandagatla. *Survey and Taxonomy of Grid Resource Management Systems.* University of Texas Austin http://www.cs.utexas.edu/users/browne/cs395f2003/projects/KandagatlaRepor.pdf, 2003.

[KCV04] I. Foster J. Frey S. Graham I. Sedukhin D. Snelling S. Tuecke K. Czajkowski, D. Ferguson and W. Vambenepe. *The WS-Resource Framework Version 1.0.* IBM, May 2004. http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.

[KJM00] Krzysztof Palacz Kyungkoo Jun, Ladislau Boloni and Dan C. Marinescu. *Agent-Based Resource Discovery.* IEEE, 2000.

[KK06] N.C. Narendra K. Kalapriya, S.K.Nandy. *A Framework for Measurement of End-To-End Qos Requirements in Loosely Coupled Systems.* 20th IEEE International Conference on Advanced Information Networking and Applications, ISBN: 1550-445X, 2006.

[KKM01] Rajkumar Buyya Klaus Krauter and Muthucumaru Maheswaran. *A taxonomy and survey of grid resource management systems for distributed computing.* John Wiley & Sons, Ltd., 2001.

[KML05] Seung-Phil Her Dong-Ryeol Shin Kyu Min Lee, Kee-Hyun Choi. *Matchmaking Algorithms to improve Dynamic Service Mathching in Ubiquitous Environments.* Fourth Annual ACIS International Conference on Computer and Informatin Science, Las Vegas, USA, 2005.

[KSL99] Seth Widoff Kaitia Sycara, Matthias Klusch and Jianguo Lu. *Dynamic service matchmaking among agents in open information environments.* SIGMOD, Volume 28, Issue 1, pp 47-53, 1999.

[KV05] Ronnie Belmans Koen Vanthournout, Geert Deconinck. *A Taxonomy for Resource Discovery.* Journal of Personal and Ubiquitous Computing, ISSN: 1617-4909, Volume 9, Issue 2, 2005.

[KW84] C. Kruskal and A. Weiss. *Allocating independent subtasks on parallel processors.* IEEE Transactions on Software Engineering, 1984.

[LC03] Li Layuan Li Changln. *An Agent-Based Approach for Grid Computing.* International Conference on Parallel and Distributed Computinge Applications and Technologies, China, pp 608-611, 2003.

[LG07] A Akram-D Colling J Martyniak M Krznaric L Guo, A S McGough. *Enabling QoS for Service-Oriented Workflow on GRID.* 7th IEEE International Conference on Computer and Information Technology, ISBN: 0-7695-2983-6, pp 1077-1082, 2007.

[LOB05] Felix Heine-Matthias Hovestadt Odej Kao Axel Keller Lars-Olof Burchard, Barry Linnert. *A Quality-of-Service Architecture for Future Grid Computing Applications.* 19th IEEE International Parallel and Distributed Processing Symposium, ISBN: 1530-2075, 2005.

[LPN04] Yueqin Jiang Jie Song Appie Stoelwinder Liang Peng, Simon See and Hoon Kang Neo. *Performance Evaluation in Computational Grid Environments.* 7th International Conference on High Performance Computing and Grid in Asia Pacific Region, ISBN: 0-7695-2138-X, 2004.

[Man93] M.Morris Mano. *Computer System Architecture, Third Edition.* Prentice Hall, 1993.

[mfws] OWL-S: Semantic markup for web services. http://www.daml.org/services/owl-s/1.0/.

[MHBL99] T. Leighton M. Harchol-Balter and D. Lewin. *Resource Discovery in Distributed Networks*. 18th Annual Symposium on Principles of Distributed Computing, ACM-SIGACT/SIGOPS, Atlanta, pp 229-238, May 1999.

[Mic05] Sun Microsystems. *N1 Grid Engine 6 Administration Guide*. Sun Microsystems, Inc. 4150 Network Circle Santa Clara, CA 95054 U.S.A., May 2005.

[MIH04] Gareth Taylor Malcolm Irving and Peter Hobson. *Plug in to Grid-Computing*. IEEE Power and Energy Magazine, ISBN: 1540-7977, pp 40-44, 2004.

[MJKB00] Hanan L. Lutyya Michael J. Katchabaw and Michael A. Bauer. *A Quality of Service Management Testbed*. The University of Western Ontario, 2000.

[MK00] M. Maheswaran and K. Krauter. *A Parameter-based approach to resource discovery in Grid computing systems*. 1st International Workshop on Grid Computing, IEEE/ACM, 2000.

[ML05] Mark Baker Maozhen Li. *The Grid Core Technologies*. John Wiley & Sons Ltd., The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, ISBN: 0470094176, 2005.

[MLBLGS03] Yannis A. Dimitriadis Miguel L. Bote-Lorenzo and Eduardo Gomez-Sanchez. *Grid Characteristics and Uses: a Grid Definition*. 1st European Across Grids Conference, Santiago de Compostela, Spain, pp 291-298, 2003.

[MPS02] T. Payne M. Paolucci, T. Kawmura and K. Sycara. *Semantic Matching of Web Services Capabilities*. 2002.

[NNN98] O. Regev N. Nisan, S. London and N.Camiel. *Globally Distributed computation over the Internet: The POPCORN project*. International Conference on Distributed Computing Systems, 1998.

[NS02] T.Y. Liu N.H. Sun. *Grid Enabling Clusters*. Journal of Computer Research and Development, ISBN: 0-262-07118-5, pp 917-922, 2002.

[Pel96] A. Pelc. *Fault-tolerant Broadcasting and Gossiping in Communication*. Networks, 1996.

[Pra08] C.S.R. Prabhu. *Grid and Cluster Computing*. Prentic-Hall of India Private Limited, ISBN: 978-81-203-3428-1, 2008.

[QC02] Umeshwar Martin Griss Qiming Chen, Meichun Hsu. *Multi-agent Cooperation, Dynamic Workflow and XML for E-commerce Automation*. HP Labs, USA, 2002.

[Qui94] M J Quinn. *Parallel Computing Theory and Practices, International Edition*. McGrawHill, 1994.

[QZ05] Jiangang Shen Chunming Rong Quan Zhou, Geng Yang. *A Scalable Security Architecture for Grid*. IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies, 2005.

[Raj98] Horowitz Sahni Rajasekaran. *Fundamentals of Computer Algorithms*. Galgotia publications, 1998.

[Ram01] Rajesh Raman. *Matchmaking Frameworks for Distributed Resource Management*. PhD Thesis University Of Wisconison Madison, 2001.

[Ram04] L. Ramakrishnan. *Securing Next Generation Grids*. IEEE Computer Society, IT Professional, Volume 6, Issue 2, 2004.

[RBD00] Steve Chapin Rajkumar Buyya and David DiNucci. *Architectural Models for Resource Management in the Grid*. Springer Verlag LNCS Series Germany, December 2000.

[RdM06] Tania Gomes Ramos and Alba Cristina Magalhaes Alves de Melo. *An Extensible Resource Discovery Mechanism for Grid Computing Environments*. IEEE International Symposium on Cluster Computing and the Grid ISBN: 0-7695-2585-7, 2006.

[RR98] Marvin Solomon Rajesh Raman, Miron Livny. *Matchmaking: Distributed Resource Management for High Throughput Computing*. The Seventh International Symposium on High Performance Distributed Computing, IEEE, 1998. 28-31 July, Chicago, Illinois, USA.

[SAL02] Peter van Santen Simone A. Ludwig. *A Grid Service Discovery Matchmaker based on Ontology Description*. EuroWeb - The Web and the Grid: from e-science to e-business, 2002.

[SB03] Allen Van Gelder Sara Base. *Computer Algorithms*. Pearson Education, ISBN: 81-7808-171-7, 2003.

[SB05] Kuldip Singh Savina Bansal, Padam Kumar. *Dealing with heterogeneity through limited duplication for scheduling precedence constrained task graphs*. Journal of Parallel and Distributed Compututing, Volume 65, Issue 4, pp 479-491, 2005.

[SB07] Sarbjeet Singh and Seema Bawa. *A Privacy, Trust and Policy Based Authorization Framework for Services in Distributed Environments*. International Journal of Computer Science, Volume 2, No. 2, pp. 85-92, 2007.

[SB08] Anju Sharma Seema Bawa. *Comparative Analysis of Resource Discovery Approaches in Grid Computing.* Journal of Computers, Vol. 3, No. 5, 2008.

[SBM04] Richard Staehli Sharath Babu Musunoori, Frank Eliassen. *QoS-Aware Component Architecture Support for Grid.* IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004.

[SD05] Jiubin Ju Liang Hu Shunli Ding, Jingbo Yuan. *A Heuristic Algorithm for Agent-based Grid Resource Discovery.* International Conference on e-Technology, e-Commerce and e-Service on e-Technology, e-Commerce and e-Service, IEEE, ISBN: 0-7695-2274-2, pp 222-225, 2005.

[SET08] SETI@Home. *More Information for it can be founded from, http://setiathome.ssl.berkeley.edu/.* University of California, 2008.

[SHL88] S. Hedetniemi S. Hedetniemi and A. Liestman. *A Survey of Gossiping and Broadcasting in Communication Networks.* Networks, 1988.

[Sie01] Sierra.C. *Agent Mediated Electronic Commerce Scientific and Technological roadmap.* The European Agent link perspective Volume 1991 of lecture notes in Artificial Intelligence, Springer-verlag, pp 1-18, 2001.

[Sig05] Kamana Sigdel. Resource allocation in heterogeneous and dynamic networks. Master's thesis, Delft University of Technology, 2005.

[SK00] D. Peleg S. Kutten. *Deterministic distributed resource discovery.* Nineteenth Annual Symposium on Principles of Distributed Computing, ACM SIGACT/SIGOPS, 2000. Portland, Oregon.

[SL02] P. Santen S. Ludwig. *A Grid Service Discovery Matchmaker based on Ontology Description.* 2002.

[SSWL02] K. Sycara, M. Klusch S. Wido, and J. Lu. *LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace.* Autonomous Agents and Multi-Agent Systems, pp 173-203, 2002.

[STY05] Kenji Kise Hiroki Honda Sanya Tangpongprasit, Takahiro Katagiri and Toshitsugu Yuba. *A time-to-live based reservation algorithm on fully decentralized resource discovery in Grid computing.* Journal of Paraller Computing, Elsevier, 2005.

[SV04] Rajkumar Buyya Akshay Luther Srikumar Venugopal, Rajiv Ranjan. *Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids.* The University of Melbourne, 2004.

[Tea01] Globus Team. *Globus Web Page.* http://www.globus.org, 2001.

[UB07] Roshan Kulkarni Umesh Bellur. *Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching.* IEEE International COnference on Web Services, Marriott Salt Lake City, Utah, USA, 2007.

[VG00] A. Aziz V. Garg. *An Efficient Deterministic Algorithm for the Resource Discovery Problem.* ECE Technical Report TR-PDS, 2000.

[VI04] Michael J. Lewis Nael B. Abu-Ghazaleh Vishal Iyengar, Sameer Tilak. *Non-Uniform Information Dissemination for Dynamic Grid Resource Discovery.* 3rd IEEE International Symposium on Network Computing and Applications, ISBN: 0-7695-2242-4, 2004.

[VS07] Neela Narayanan V and Kailash S. *Resource Matchmaking in Grid*

*-Semantically*. 9th IEEE International Conference on Advanced Communication Technology, ISBN: 978-89-5519-131-8 93560, pp 2051-2055, 2007.

[Wat01] Steven Waterhouse. Jxta search: Distributed search for distributed networks, 2001.

[Wen04] Horst Wenske. Condor clusters with multilateral resource matchmaking in heterogenous networks, 2004.

[WL02] Fangpeng Dong Jun Zhang Wei Li, Zhiwei Xu. *Grid Resource Discovery Based on a Routing-Transferring Model*. Third International Workshop on Grid Computing, Springer-Verlag, London, UK, pp 145-156, 2002.

[YAT99] M. Factor Y. Aridor and A. Teperman. *cJVM: a Single System Image of a JVM on a Cluster*. 29th International Conference on Parallel Processing, Fukushima, Japan, IEEE Computer Society Press, USA, 1999.

[YH02] N. Venkatasubramanian Y. Huang. *QoS-based resource discovery in intermittently available environments*. 11th International Symposium on High Performance Distributed Computing, IEEE, pp 50 -59, 2002.

[Zhu02] H. Zhuge. *A Knowledge Grid Model and Platform for Global Knowledge Sharing*. Expert Systems with Applications, pp 313-320, 2002.

[ZX02a] G. Mei Z. Xu, X. Li. *The Research on Architecture of Vega Information Grid*. Journal of Computer Research and Development, pp 948-951, 2002.

[ZX02b] W. Li Z. Xu. *The Research on Vega Grid Architecture*. Journal of Computer Research and Development, pp 923-929, 2002.

[ZX02c] Hongguang Fu Zhenbing Zeng Zhiwei Xu, Wei Li. *The Vega Grid and Grid-Based Education*. 1st International Conference on Web-Based Leaming, HK, China, pp 228-240, 2002.

# List of Publications

## Journal Publications

(1.) Anju Sharma, Seema Bawa, "An Improved Resource Discovery Approach Using P2P Model for Condor: A Grid Middleware", Transactions on Engineering, Computing and Technology, World Enformatika Society, ISSN 1305-5313, Volume 17, December 2006, pp 55-59.

(2.) Anju Sharma, Seema Bawa, "Optimized Resource Discovery Algorithm for Grid Computing", International Journal of Computational Science, Global Information Publisher, Volume 2, Number 1, February 2008, ISSN 1992-6669 (Print) ISSN 1992-6677 (Online), pp 157-165.

(3.) Anju Sharma, Seema Bawa, "Comparative Analysis of Resource Discovery Approaches in Grid Computing", Journal of Computer Science, Academy Publisher, ISSN 1796-203X, Volume 3, Number 5, May 2008, pp 60-64.

(4.) Anju Sharma, Seema Bawa, "TUGrid Portal: A Secure and User-friendly Interface" International Transactions on Systems Science and Applications, ISSN 1751-1461(Print), ISSN 1751-147X (CD-ROM), Volume 4, Number 2, May 2008, pp. 188-193.

## Conference Publications

(5.) Anju Sharma, Seema Bawa, "QoS Integrated Resource Discovery in a Campus Grid: A Case Study", ICISTM 2007: International Conference on Information Systems, Technology and Management, New Delhi, India, March 12-13, 2007, pp 75-81.

(6.) Seema Bawa, Anju Sharma, "A Deadlock Detection Algorithm for Grid Resource Management", CMMSE 2007: 7th International Conference on Computational and Mathematical Methods in Science & Engineering, Illinois Institute of Technology, Chicago, Illinois, USA, June 20-23, 2007, pp 52-58.

(7.) Anju Sharma, Seema Bawa, "Portal for the Computational Grid: TU Grid Portal", HiPAAC 2007: International Conference on Emerging Trends in High Performance Architecture Algorithms & Computing, SSN Campus, Chennai, India, July 12-13, 2007, pp 218.

(8.) Anju Sharma, Seema Bawa, "A Framework for Resource Discovery in Grid Environment", ICDEM 2008: International Conference on Data Engineering and Management, Bishop Heber College, Tiruchirappalli, India, February 9, 2008, pp 195-198.

# Appendix-A: List of Abbreviations

1. VOs: Virtual Organizations
2. SETI : Search for Extraterrestrial Intelligence
3. QoS: Quality of Service
4. GRAM: Grid Resource Allocation Manager
5. GridFTP: Grid File Transfer Protocol
6. GRIS: Grid Resource Information Service
7. IP: Internet Protocols
8. TCP/IP: Transfer Control Protocol/Internet Protocols
9. DNS: Denial of Service
10. ICMP: Internet Control Message Protocol
11. UDP: User Datagram Protocol
12. SSL: Secure Socket Layer
13. APIs: Application Program Interfaces
14. RSL: Resource Specification Language

15. IT: Information Technology
16. ClassAds: classified ads
17. SLA: Service Level Agreement
18. ATM: Asynchronous Transfer Mode
19. LDAP: Light Wight Directory Access Protocol
20. P2P: Peer-to-Peer
21. SDRT: Shortest Distance Routing Transferring
22. DIAR: Discovering Intermittently Available Resources
23. GRAM: Grid Resource Allocation Manager
24. GASS: Global Access to Secondary Storage
25. GSI: Grid Service Infrastructure
26. MDS: Meta Directory Service
27. GIIS: Grid Index Information Services
28. FCFS: First Come First Serve
29. HTC: High Throughput Computing
30. schedd: Schedular Daemon
31. commd: Communication Daemon
32. shadowd: Shadow Daemon
33. RLA: Resource Level Agreement
34. TSLA: Task Service Level Agreement

35. RSLA: Resource Level Agreement
36. BSLA: Binding Service Level Agreement
37. GARA: Globus Architecture for Reservation and Allocation
38. SU: Subject
39. SR: Service
40. R: Resource
41. DO: Domain
42. AC: Access
43. CM: Central Manager
44. MST: Minimum Spanning Tree
45. WPP: Weightage Per Processor
46. $Comm_{ij}$: Communication Cost
47. $Comp_i$: Computation Cost
48. CSED: Computer Science and Engineering
49. ECED: Electronics and Communication Engineering
50. EIED: Electrical and Instrumentation Engineering
51. MED: Mechanical Engineering
52. DBTS: Biotechnology and Environmental Sciences
53. CED: Chemical Engineering
54. SMCA: Mathematics and Computer Application

55. SCBC: Chemistry and Biochemistry

56. SPMS: Physics and Material Sciences

57. SMSS: Management and Social Sciences

58. LMTSM: L. M. Thapar School of Management

59. TUGrid: Thapar University Grid