Semantic Discovery of Services in a Grid Environment

A thesis
submitted in partial fulfillment of the requirements
for the award of degree
of

Master of Engineering
in
Software Engineering

By:
Shruti Goel
(Roll No. 8043123)

Under the supervision of:
Ms. Inderveer Chana
Senior Lecturer
CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147004

MAY 2006

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Semantic Discovery of Services in a Grid Environment"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Ms. Inderveer Chana.

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

(Shruti Goel)

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

(Ms. Inderveer Chana)
Supervisor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
Patiala

Countersigned by

(Dr.(Mrs) Seema Bawa)
Head
Computer Science & Engineering Department
Thapar Institute of Engg. and Tech.,
Patiala.

(Dr. T. P. Singh)
Dean
Academic Affairs
Thapar Institute of Engg. and Tech.,
Patiala

Acknowledgement

I wish to express my deep gratitude to Ms. Inderveer Chana, Senior Lecturer, Computer Science & Engineering Department for providing her uncanny guidance and support throughout the preparation of the thesis report.

I am thankful to Dr. (Mrs.) Seema Bawa, Head, and Mr. Rajesh Bhatia, P.G. Coordinator, Computer Science & Engineering Department, for the motivation and inspiration that triggered me for the thesis work.

I am also thankful to Mr. Maninder Singh, Assistant Professor, Computer Science & Engineering Department, for providing timely assistance and encouragement that went a long way in successful completion of my thesis.

I would also like to thank all the staff members, T.I.E.T. Grid Group and all my friends especially Rajiv Bammi and Meena Gupta who were always there at the need of the hour and provided all the help and facilities, which I required for the completion of the thesis.

Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Shruti Goel
(8043123)

Abstract

Grid is an emerging technology for enabling resource sharing and coordinated problem solving in dynamic multi-institutional virtual organizations. In the Grid environment, the resources may belong to different institutions, have different usage policies and pose different requirements on acceptable requests. One of the fundamental operations needed to support location-independent computing is resource discovery. Before resources can be allocated to run an application, a user or agent must discover resources appropriate to the requirements of the application. This process of locating relevant resources based on application requirements is called resource discovery.

The ability to describe these Grid resources and services needed by applications is essential for providing seamless access to them on the Grid. However, as yet there is no universal resource description language common to all Grid middleware systems. Different Grid middleware systems have different ad hoc methods of resource description and it is not yet known how well these can interoperate. Hence, there is a need to utilize semantic matching of these resource descriptions, firstly because there is currently no common standard, and secondly to make the Grid transparent at the application level.

In this thesis work, we have setup a grid environment and then implemented a semantic based search approach to discover the resources and services similar to the ones requested by the user. Semantic search methods seek to augment and improve traditional search results by finding services that have similar concepts not just similar descriptions. Using semantic information for the matchmaking process achieves better results than exact keyword matching prevalent in most of the resource discovery approaches used today.

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter gives an overview of the work done in this thesis. It focuses on a brief introduction of grid computing and how services are discovered in a grid environment. Then we discuss the problem found in existing approaches for service discovery and the solution we have implemented. Finally this chapter gives the organization of the thesis along with a brief idea about the contents of each of the following chapters.

1.1 What is Grid Computing?

Grid computing [10,12], most simply stated, is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various resources. Grid computing [32] spans multiple organizations, machine architectures and software boundaries to provide unlimited power, collaboration and information access to everyone connected to a grid. In other words, Grid is a network of heterogeneous resources working in collaboration to solve problems that cannot be addressed by the resources of any one organization.

1.2 Defining the Problem

The first requirement for successful sharing of resources among the various organizations is an efficient discovery mechanism for locating the desired resources available in the grid system at time of request, however, due to the dynamic, heterogeneous and distributed nature of the Grid environment, resource discovery is a difficult job. Grid resources are potentially very large in number and variety; individual resources are not centrally controlled, and they can enter and leave the grid systems at any time. For these reasons, resource discovery in large-scale grids can be very challenging.

In the OGSA framework each resource is represented as a Grid service, therefore resource discovery mainly deals with the problem of locating and querying information about useful Grid services. Thus, henceforth we will be using the terms resource and service interchangeably.

Currently service discovery is done based on symmetric, attribute-based keyword matching. In such systems, the values of attributes advertised by services are compared for exact match with those required by jobs. For the comparison to be meaningful and effective, the service providers and consumers have to agree upon the syntax of attribute names and values beforehand so that the requestor has prior knowledge about the description of a particular service. However, in a heterogeneous multi-institutional environment such as the Grid which is setup using a number of middleware, it is difficult to enforce the syntax of service descriptions. Thus, there is a need to construct systems that are capable of fulfilling the requests intelligently and by "understanding" the terms involved and the relationships between them.

1.3 Methodology Used

Semantic matching [4,37] technique is used, because of its ability of inexact matching, to solve the problem of interoperability among the services described by various middleware. Semantic matching deals with concepts and logical relationships thus finding all those supplies that can fulfill a demand even to certain extent. It is quite different from traditional approaches that look for exact match between resource description and job request. Instead, semantic matching uses its knowledge and its semantic understanding of the advertisement and request to find all those resources that are similar to the required resource even if they do not match exactly. Using semantic information for the matchmaking [36,44] process achieves better results than syntactic type matchmaking.

1.4 Organization of Thesis

The chapters in the thesis are organized as follows:

Chapter 2 describes in detail what grid computing is, how it evolved from distributed computing, various types of grid systems, grid architecture, the open standards used by grid and benefits of grid.

Chapter 3 describes the resource discovery process and various approaches and algorithms used by this process.

Chapter 4 discusses the problems found in the existing approaches to resource discovery and the possible solutions to these problems.

Chapter 5 includes the implementation details of the grid setup and semantic based search engine for service discovery in the grid environment.

Chapter 6 summarizes the work presented in this thesis followed by the features that can be incorporated in future for the enhancement of the Semantic Search Engine.

Chapter 2

Grid Computing Concepts

This chapter describes grid computing in detail. It explains the evolution of grid computing from distributed computing, various kinds of grids based on the kind of service they provide, different topologies of the grid depending on the number of organizations that are a part of a particular grid environment and benefits of a grid environment. It also gives an insight into the architecture of the grid and service oriented architecture on which the grid architecture is based. Then it explains the open standard OGSA specified for standard-based grid computing.

2.1 Introduction to Grid Computing

Grid computing [14,15] is the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. It is a hardware and software infrastructure that clusters and integrates high-end computers, networks, databases and scientific instruments from multiple sources to form a virtual supercomputer on which users can work collaboratively.

Web is a service for sharing information over the Internet, similarly grid [32] is a service for sharing computer power and data storage capacity over the Internet. The Grid goes well beyond simple communication between computers and aims ultimately to turn the global network of computers into one vast computational resource.

The goal of grid computing is to realize a persistent, secure, efficient, standards-based service infrastructure that enables coordinated sharing of autonomous and geographically distributed hardware, software, and information resources in dynamic, multi-institutional virtual organizations. The sharing in this context is not primarily file exchange, as supported by the Web or peer-to-peer systems, but rather direct access to computers, software, data, services, and other resources, as required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and

engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization (VO).

2.2 Virtual Organization

The Grid is *"flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources—what we refer to as virtual organizations."* Ian Foster et al [15], 2001.

A virtual organization [13,15] can be defined as an organization providing virtual resources in a way that the users or customers do not have to know the inner goal, structure, and rules of the organization, but inside the organization a clear manageable functional and process organization exists. The management inside the organization is not in conflict with the dynamism of its elements. An ideal virtual organization would provide ideal transparent utilization of their resources.

2.3 History of Grid

It all started with Distributed computing [33]. The development of Grid computing [10,12,14] technology that allows resource sharing across multiple domains is driven by the need for large-scale high performance distributed computing. A distributed system consists of a set of software agents that work together to implement some intended functionality. Because the agents in a distributed system do not operate in a uniform processing environment, they must communicate by protocol stacks that are intrinsically less reliable than direct code invocation and shared memory. Grid-computing principles focus on large-scale resource sharing in distributed systems in a flexible, secure, and coordinated fashion. This dynamic coordinated sharing results in innovative applications making use of high-throughput computing for dynamic problem solving. Grid computing [12,15] aims to resolve the complexities of distributed computing through a well-defined architecture that is aligned with SOA (Service Oriented Architecture), autonomic-computing principles, and open standards for integration and interoperability [23].

The basic idea behind the Grid is sharing computing power. Nowadays most people have more than enough computing power on their own PC, hence sharing is unnecessary for most purposes. However back in the sixties and seventies, sharing computer power was essential. At that time, computing was dominated by huge mainframe computers, which had to be shared by whole organizations. A number of applications needed more computing power than can be offered by a single resource or organization in order to solve them within a feasible/reasonable time and cost. This promoted the exploration of logically coupling geographically distributed high-end computational resources and using them for solving large-scale problems. Such emerging infrastructure was called computational grid, and led to the popularization of a field called grid computing.

Back in 1965 the developers of an operating system called Multics (an ancestor of Unix) presented a vision of "computing as an utility" - in many ways uncannily like the Grid vision today. Access to the computing resources was envisioned to be exactly like access to utility such as electricity - something that the client connects to and pays for according to the amount of use. This led to coining of the term “Grid” [27]. As electrical grids provide consistent, pervasive, dependable, transparent access to electricity irrespective of its source, similarly computing grids offer services to users without letting them know where the source is located. Analogous to the power grid, grid environments create virtual providers, carriers and consumers to avoid single points of failure, and deliver a continuous stream of computing power.

2.4 Grid Systems Taxonomy

Depending upon the design objectives and target applications for a particular Grid environment grid systems can be classified into three categories as shown in figure 2.1. The design objective of a grid system can be any one or a combination of two or more of the following.

- improving application performance,
- improving data access, and
- enhanced services.

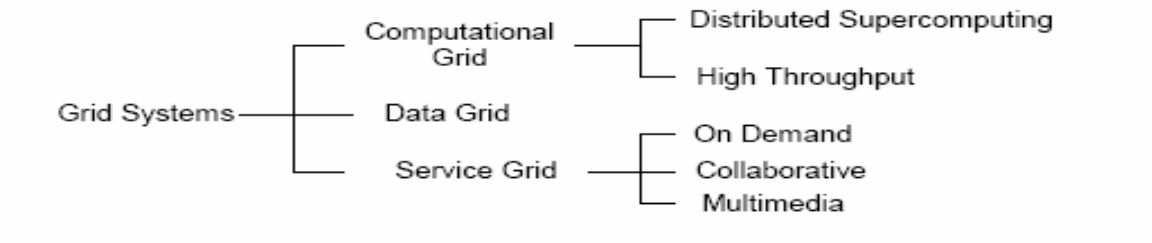

Figure 2.1 Grid Systems Taxonomy [34]

The **computational Grid** [12,34] category denotes systems that have a higher aggregate computational capacity available for single applications than the capacity of any constituent machine in the system. These can be further subdivided into distributed supercomputing and high throughput categories depending on how the aggregate capacity is utilized.

- A **distributed supercomputing** Grid executes the application in parallel on multiple machines to reduce the completion time of a job. Typically, applications that require a distributed supercomputer are grand challenge problems. Examples of grand challenge problems are fluid dynamics, weather modeling, nuclear simulation, molecular modeling, and complex financial analysis.
- A **high throughput Grid** increases the completion rate of a stream of jobs. Applications that involve parameter study to explore a range of possible design scenarios such as ASIC or processor design verification tests would be run on a high throughput Grid.

The **data Grid** [34] category is for systems that provide an infrastructure for synthesizing new information from data repositories such as digital libraries or data warehouses that are distributed in a wide area network. Computational Grids also need to provide data services but the major difference between a Data Grid and a computational Grid is the specialized infrastructure provided to applications for storage management and data access. In a computational Grid the applications implement their own storage management schemes rather than use Grid provided services. Typical applications for these systems would be special purpose data mining that correlates information from multiple different data sources and then processes it further. The two popular DataGrid

initiatives, European DataGrid [9] and Globus [10,19], are working on developing large-scale data organization, catalog, management, and access technologies.

The **Service Grid** [34] category is for systems that provide services that are not provided by any single machine. This category is further subdivided in on demand, collaborative, and multimedia Grid systems.

- A **Collaborative Grid** connects users and applications into collaborative workgroups. True collaboration will enable organizations to work on common problems regardless of their geographic locations. These systems enable real time interaction between humans and applications via a virtual workspace.
- A **Multimedia Grid** provides an infrastructure for real-time multimedia applications. This requires supporting quality of service across multiple different machines whereas a multimedia application on a single dedicated machine can be deployed without QoS.
- An **On demand Grid** facilitates access to rarely used resources for short periods of time. While many organizations and individuals would benefit from resources such as a supercomputer or specialized instrumentation equipment, their infrequent usage pattern would fail to justify the procurement of such a resource. Grid services can allow seamless use of these resources, regardless of the given domain.

2.5 Grid Topologies

There are three topologies of grid namely, Intragrid, Intergrid and Extragrid. The simplest of these is the intragrid [10], which is comprised merely of a basic set of grid services within a single organization. The complexity of the grid design is proportionate to the number of organizations that the grid is designed to support, and the geographical parameters and constraints. As more organizations join the grid, the non-functional or operational requirements for security, directory services, availability, and performance become more complex. The topological view is shown in figure 2.2.

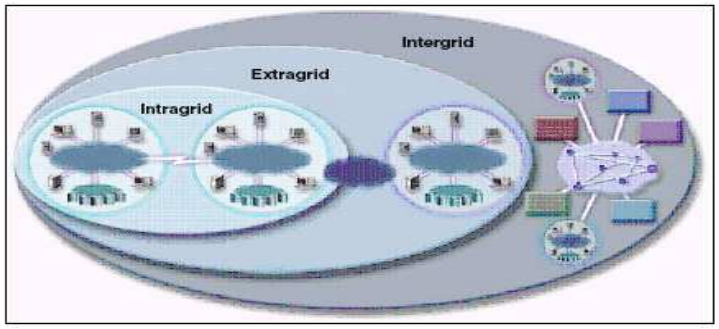

Figure 2.2. Topological view of Grids [10]

2.5.1 Intragrid

A typical intragrid topology, as illustrated in figure 2.3 below, exists within a single organization, providing a basic set of grid services. The single organization could be made up of a number of computers that share a common security domain, and share data internally on a private network. The primary characteristics of an intragrid are a single security provider, bandwidth on the private network is high and always available, and there is a single environment within a single network. Within an intragrid, it is easier to design and operate computational and data grids. An intragrid provides a relatively static set of computing resources and the ability to easily share data between grid systems.

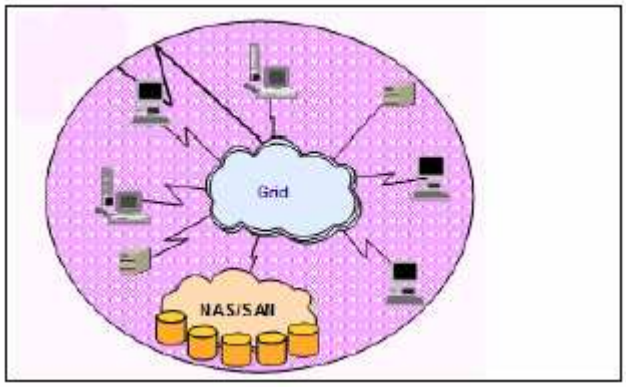

Figure 2.3. Intragrid [10]

2.5.2 Extragrid

Based on a single organization, the extragrid [10] expands on the concept by bringing together two or more intragrids. An extragrid, as illustrated in the figure 2.4 given below, typically involves more than one security provider, and the level of management

complexity increases. The primary characteristics of an extragrid are dispersed security, multiple organizations, and remote/WAN connectivity. Within an extragrid, the resources become more dynamic and the grid needs to be more reactive to failed resources and failed components. The design becomes more complicated and information services [8,34] become relevant to ensure that grid resources have access to workload management at run time.

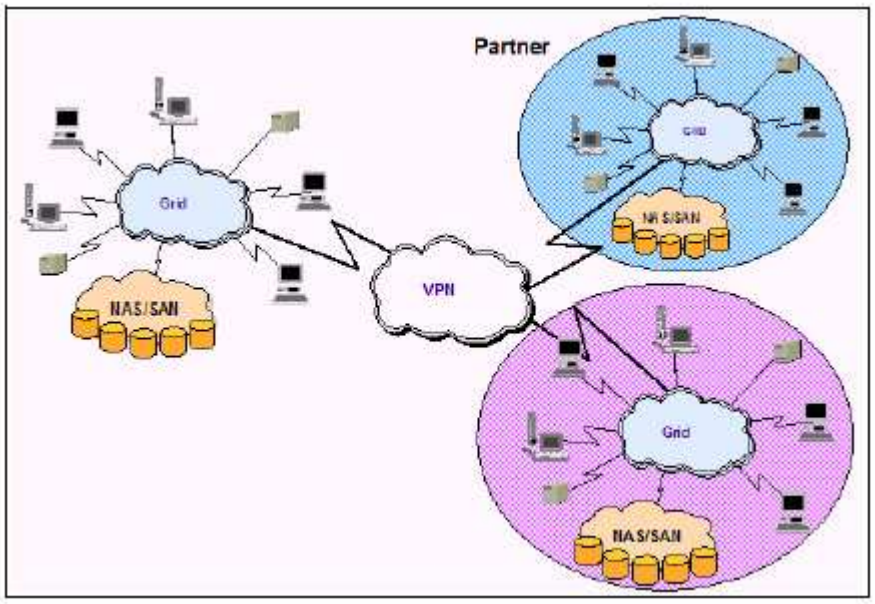

Figure 2.4. Extragrid [10]

2.5.3 Intergrid

An intergrid [10] requires the dynamic integration of applications, resources, and services with patterns, customers, and any other authorized organizations that will obtain access to the grid via the internet/WAN. An intergrid topology, as illustrated in figure 2.5, is primarily used by engineering firms, life science industries, manufacturers, and by businesses in the financial industry. The primary characteristics of an intergrid include dispersed security, multiple organizations, and remote/WAN connectivity. The data in an intergrid is global public data, and applications, both vertical and horizontal, must be modified for a global audience.

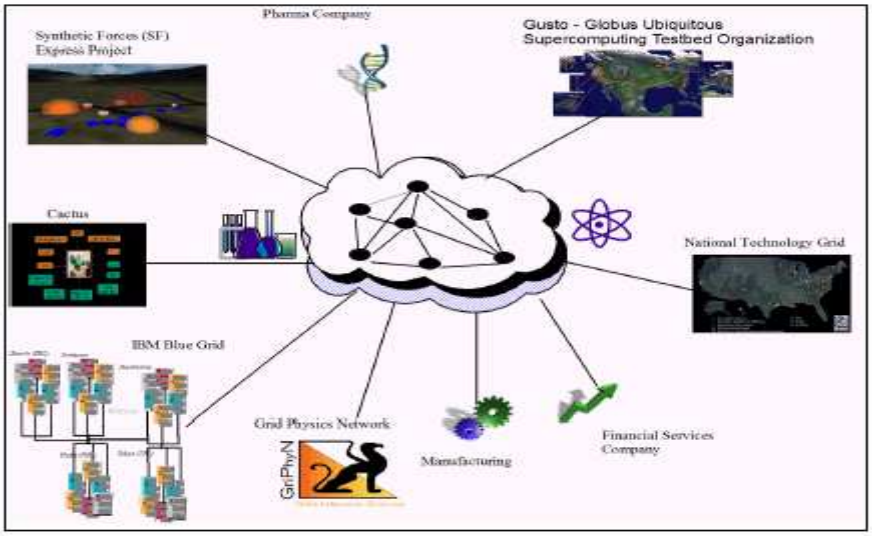

Figure 2.5 Intergrid [10]

2.6 The Grid Architecture

The grid architecture [33] identifies the basic components of a grid system. It defines the purpose and functions of its components, while indicating how these components interact with one another. The main focus of the grid architecture is on interoperability among resource providers and users in order to establish the sharing relationships. This interoperability, in turn, necessitates common protocols at each layer of the architectural model, which leads to the definition of a grid protocol architecture as shown in the following figure.

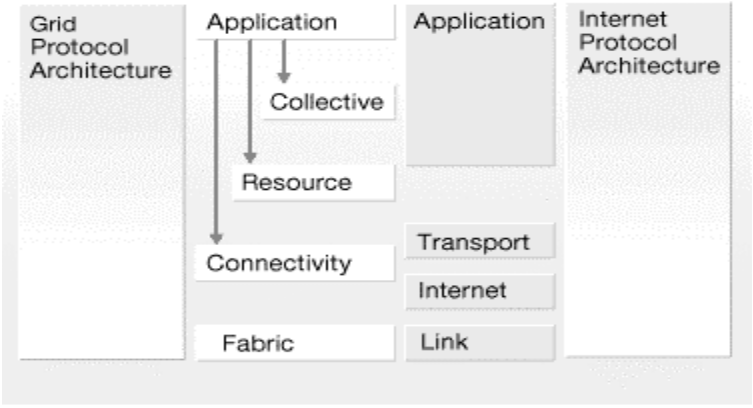

Figure 2.6 The layers of the grid Architecture and its relationship to the Internet Protocol Architecture [33]

This protocol architecture defines common mechanisms, interfaces, schema, and protocols at each layer, by which users and resources can negotiate, establish, manage, and share resources. Figure 2.6 shows the component layers of the grid architecture and the capabilities of each layer. Each layer shares the behavior of the underlying component layers. The following describes the core features of each of these component layers, starting from the bottom of the stack and moving upward.

Fabric layer—The fabric layer [33] defines the interface to local resources, which may be shared. This includes computational resources, data storage, networks, catalogs, software modules, and other system resources.

Connectivity layer—The connectivity layer [33] defines the basic communication and authentication protocols required for grid-specific networking-service transactions.

Resource layer—This layer uses the communication and security protocols (defined by the connectivity layer) to control secure negotiation, initiation, monitoring, accounting, and payment for the sharing of functions of individual resources. The resource layer [33] calls the fabric layer functions to access and control local resources. This layer only handles individual resources, ignoring global states and atomic actions across the resource collection pool, which are the responsibility of the collective layer.

Collective layer—While the resource layer manages an individual resource, the collective layer [33] is responsible for all global resource management and interaction with collections of resources. This protocol layer implements a wide variety of sharing behaviors using a small number of resource-layer and connectivity-layer protocols.

Application layer—The application layer [33] enables the use of resources in a grid environment through various collaboration and resource access protocols.

2.7 Service Oriented Architecture (SOA)

The grid architecture can be merged with emerging architectures such as the service oriented architecture (SOA) so that the grid can quickly adapt to a wider technology domain.

SOA is a specific type of distributed system framework that maintains agents that act as "software services," performing well-defined operations. It is a loosely coupled architecture with a set of abstractions relating to components granular enough for consumption by clients and accessible over the network with well-defined policies as dictated by the components.

Thus, a service acts as a user-facing software component of an application or resource. This paradigm of functionality enables the users of that application (or resource pool) to be concerned only with the operational description of the service. In addition, SOA stresses that all services have a network-addressable interface and communicate via standard protocols and data formats called messages.

The Web Services Architecture (WSA) [33] helps to enable and define SOA, where services interact by exchanging XML (Extensible Markup Language) messages, as shown in Figure 2.7 given below.

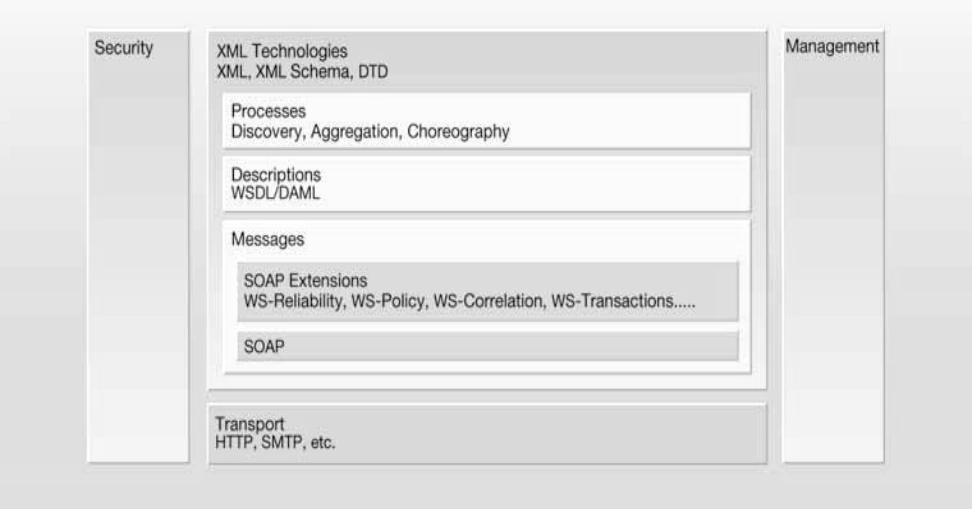

Figure 2.7. The Web Services Architecture (WSA) and the respective layers in the WSA Framework [33]

The main characteristics of the WSA are:

- It is based on XML technologies such as XML Information Model, XML Base, and XML Schema.
- It is independent of the underlying transport protocols (e.g., HTTP or SMTP) and the transport selected as a runtime binding mechanism.
- It uses XML message exchanges, while providing the extensibility model for this message exchange pattern to adapt to various message interchange requirements (e.g., security, reliability, correlation and privacy.) These interoperable messages could be created using Simple Object Access Protocol (SOAP) [3,47] and SOAP extensions. SOAP extension mechanisms and XML information models are used to construct Web services extensions.
- Service capabilities are described using description languages such as Web Services Description Language (WSDL) [7,54,55].
- It uses the service discovery [26,57], workflow choreography, and transaction/state management that are built on the XML capabilities and lower-layer specifications in the architecture layer.

The emergence of the SOA concept helps grid resources to advertise their capabilities through standard interfaces defined as part of their service extensions. This enables grid users to integrate the resources through open-standards interfaces. In addition, the operational functions of each layer in the grid architecture can be abstracted to enable easier integration across all the layers. This service abstraction of each layer of the grid architecture is shown in Figure 2.8 given below.

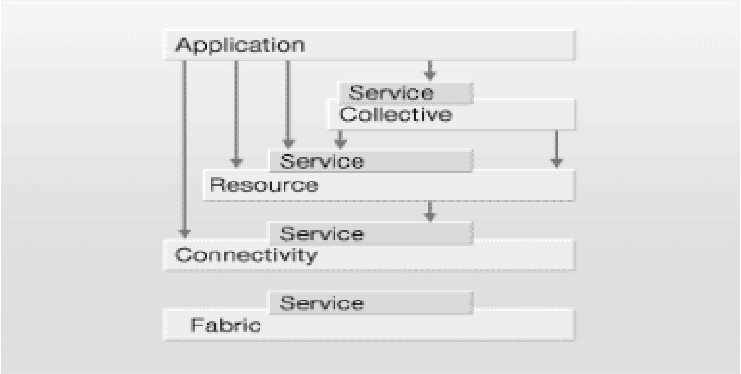

Figure 2.8. Service-oriented grid architecture with service interfaces [33]

2.8 Open Standards Platform

In order to achieve true distributed resource sharing across heterogeneous and dynamic VOs, grid-computing technologies require several improvements in alignment with other computing technologies. In the early days of grid computing, a number of custom middleware solutions were created to solve the grid problem, but this resulted in non-interoperable solutions and problematic integration among the participants. The new wave of grid computing focuses on the easier integration, security, and QoS aspects of resource sharing.

Foster et al described the Open Grid Services Architecture (OGSA) [13,16] as a solution to the above problem. This architectural concept is a result of the alignment of existing grid standards with emerging SOAs, as well as with the Web. OGSA [14,33] provides a uniform way to describe grid services and define a common pattern of behavior for these services. It defines grid-service behavior, service description mechanisms, and protocol binding information by using Web services [20] as the technology enabler. This architecture uses the best features from both the grid-computing community and the Web-services community.

"Open Grid Service Architecture" (OGSA) is the industry blueprint for standards-based grid computing. "Open" refers to both the standards-development process and the standards themselves. OGSA [16,33] is "service-oriented" because it delivers functionality among loosely coupled interacting services that are aligned with industry-accepted Web service standards. "Architecture" defines the components, their organizations and interactions, and the overall design philosophy.

OGSA is the responsibility of the Global Grid Forum (GGF), working in conjunction with other leading standards organizations engaged in delivering industry-standard distributed computing. GGF is the community of users, developers, and vendors leading the global standardization effort for grid computing.

2.9 OGSA Architecture

OGSA is a layered architecture, as shown in Figure 2.9, with clear separation of the functionalities at each layer. As seen in the figure, the core architecture layers are the Open Grid Services Infrastructure (OGSI) and OGSA platform services. The platform services establish a set of standard services including policy, logging, service level management, and other networking services. High-level applications and services use these lower-layer platform core components to become a part of a resource-sharing grid.

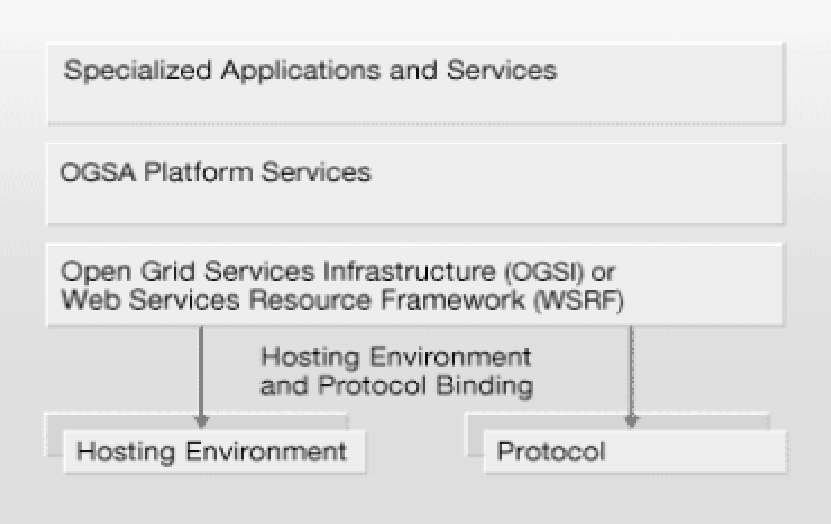

Figure 2.9.OGSA platform architecture [33]

OGSA specifies following eight high-level categories of services:

- **Infrastructure Services** enable communication between disparate resources (computer, storage, applications, etc.), removing barriers associated with shared utilization.
- **Resource Management Services** enable the monitoring, reservation, deployment, and configuration of grid resources based on quality of service requirements
- **Data Services** enable the movement of data where it is needed – managing replicated copies, query execution and updates, and transforming data into new formats if required.

- **Context Services** describe the required resources and usage policies for each customer that utilizes the grid – enabling resource optimization based on service requirements.
- **Information Services** provide efficient production of, and access to, information about the grid and its resources, including status and availability of a particular resource.
- **Self-Management Services** support the attainment of stated levels of service with as much automation as possible, to reduce the costs and complexity of managing the system.
- **Security Services** enforce security policies within a virtual organization, promoting safe resource-sharing and appropriate authentication and authorization of users.
- **Execution Management Services** enable both simple and more complex workflow actions to be executed, including placement, provisioning, and management of the task lifecycle.

2.10 Benefits of Grid Computing

This section describes the benefits of a grid environment in detail.

2.10.1 Exploiting underutilized resources

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines are busy less than 5 percent of the time. In some organizations, even the server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage.

The processing resources are not the only ones that may be underutilized. Often, machines may have enormous unused disk drive capacity. Grid computing, more specifically, a “data grid”, can be used to aggregate this unused storage into a much larger virtual data store, possibly configured to achieve improved performance and reliability over that of any single machine.

2.10.2 Parallel CPU capacity

The potential for massive parallel CPU capacity [10] is one of the most attractive features of a grid. In addition to pure scientific needs, such computing power is driving a new evolution in industries such as the bio-medical field, financial modeling, oil exploration, motion picture animation, and many others.

The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts. A CPU intensive grid application can be thought of as many smaller "subjobs," each executing on a different machine in the grid. The scalability of the application depends on the extent of communication between these subjobs. A perfectly scalable application will, for example, finish 10 times faster if it uses 10 times the number of processors.

2.10.3 Collaboration of virtual resources

Another important grid computing contribution is to enable and simplify collaboration among a wider audience. In the past, distributed computing promised this collaboration and achieved it to some extent. Grid computing takes these capabilities to an even wider audience, while offering important standards that enable very heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of virtual resources. The users of the grid can be organized dynamically into a number of virtual organizations [13,15], each with different policy requirements. These virtual organizations can then share their resources collectively as a larger grid.

Sharing starts with data in the form of files or databases. A "data grid" [34] can expand data capabilities in several ways. First, files or databases can seamlessly span many systems and thus have larger capacities than on any single system. Such spanning can improve data transfer rates through the use of striping techniques. Data can be duplicated throughout the grid to serve as a backup and can be hosted on or near the machines most likely to need the data, in conjunction with advanced scheduling techniques.

Sharing is not limited to files, but also includes many other resources, such as equipment, software, services, licenses, and others. These resources are "virtualized" to give them a more uniform interoperability among heterogeneous grid participants. The participants and users of the grid can be members of several real and virtual organizations. The grid can help in enforcing security rules among them and implement policies, which can resolve priorities for both resources and users.

2.10.4 Access to additional resources

In addition to CPU and storage resources, a grid can provide access to increased quantities of other resources and to special equipment, software, licenses, and other services. The additional resources can be provided in additional numbers and/or capacity. For example, if a user needs to increase his total bandwidth to the Internet to implement a data mining search engine, the work can be split among grid machines that have independent connections to the Internet. In this way, the total searching capability is multiplied, since each machine has a separate connection to the Internet. If the machines had shared the connection to the Internet, there would not have been an effective increase in bandwidth.

Some machines may have expensive licensed software installed that the user requires. His jobs can be sent to such machines more fully exploiting the software licenses. Some machines on the grid may have special devices. Most of us have used remote printers, perhaps with advanced color capabilities or faster speeds. Similarly, a grid can be used to make use of other special equipment. For example, a machine may have a high speed, self-feeding, DVD writer that could be used to publish a quantity of data faster. Some machines on the grid may be connected to scanning electron microscopes that can be operated remotely. In this case, scheduling and reservation are important. A specimen could be sent in advance to the facility hosting the microscope. Then the user can remotely operate the machine, changing perspective views until the desired image is captured. The grid can enable more elaborate access, potentially to remote medical diagnostic and robotic surgery tools with two-way interaction from a distance. The

variations are limited only by one's imagination. Today, we have remote device drivers for printers.

2.10.5 Reliability

High-end conventional computing systems use expensive hardware to increase reliability. They are built using chips with redundant circuits that vote on results, and contain much logic to achieve graceful recovery from an assortment of hardware failures. The machines also use duplicate processors with hot pluggability so that when they fail, one can be replaced without turning the other off. Power supplies and cooling systems are duplicated. The systems are operated on special power sources that can start generators if utility power is interrupted. All of this builds a reliable system due to the duplication of high-reliability components.

Thus, if there is a power or other kind of failure at one location, the other parts of the grid are not likely to be affected. Grid management software can automatically resubmit jobs to other machines on the grid when a failure is detected. In critical, real-time situations, multiple copies of the important jobs can be run on different machines throughout the grid. Their results can be checked for any kind of inconsistency, such as computer failures, data corruption, or tampering; thus offering much more reliability.

2.10.6 Management

The goal to virtualize the resources on the grid and more uniformly handle heterogeneous systems will create new opportunities to better manage a larger, more disperse IT infrastructure. It will be easier to visualize capacity and utilization, making it easier for IT departments to control expenditures for computing resources over a larger organization.

The grid offers management of priorities among different projects. In the past, each project may have been responsible for its own IT resource hardware and the expenses associated with it. Often this hardware might be underutilized while another project finds itself in trouble, needing more resources due to unexpected events. With the larger view a grid can offer, it becomes easier to control and manage such situations. The

administrators can change any number of policies that affect how the different organizations might share or compete for resources. Aggregating utilization data over a larger set of projects can enhance an organization's ability to project future upgrade needs. When maintenance is required, grid work can be rerouted to other machines without crippling the projects involved. Various tools may be able to identify important trends throughout the grid, informing management of those that require attention.

2.10.7 Resource Balancing

A grid federates a large number of resources contributed by individual machines into a greater total virtual resource. For applications that are grid-enabled, the grid can offer a resource balancing effect by scheduling grid jobs on machines with low utilization. This feature can prove invaluable for handling occasional peak loads of activity in parts of a larger organization. This can happen in two ways:

- An unexpected peak can be routed to relatively idle machine.
- If the grid is already fully utilized, the lowest priority work being performed on the grid can be temporarily suspended or even cancelled and performed again later to make room for the higher priority work.

Without a grid infrastructure, such balancing decisions are difficult to prioritize and execute. Occasionally, a project may suddenly rise in importance with a specific deadline. A grid cannot perform a miracle and achieve a deadline when it is already too close. However, if the size of the job is known, if it is a kind of job that can be sufficiently split into subjobs, and if enough resources are available after preempting lower priority work, a grid can bring a very large amount of processing power to solve the problem. In such situations, a grid can, with some planning, succeed in meeting a surprise deadline.

Other more subtle benefits can occur using a grid for load balancing. When jobs communicate with each other, the Internet, or with storage resources, an advanced scheduler could schedule them to minimize communications traffic or minimize the

distance of the communications. This can potentially reduce communication and other forms of contention in the grid.

Finally, a grid provides an excellent infrastructure for brokering resources. Individual resources can be profiled to determine their availability and their capacity, and this can be factored into scheduling on the grid. Different organizations participating in the grid can build up grid credits and use them at times when they need additional resources. This can form the basis for grid accounting and the ability to more fairly distribute work on the grid.

2.11 Conclusion

This chapter described in detail what grid computing is, evolution of grid computing from distributed computing, various kinds of grids based on the kind of service they provide, different topologies of the grid depending on the number of organizations that are a part of a particular grid environment, benefits of grid environment, architecture of the grid, service oriented architecture and lastly open standard OGSA specified for standard-based grid computing.

The next chapter discusses the role of information services in grid architecture, resource discovery process and its steps, and the various existing approaches and algorithms used for grid service discovery.

Chapter 3

Resource Discovery Approaches And Algorithms

This chapter describes information services which is one of the most important services provided by grid architecture. Next we discuss the resource discovery process and steps followed in this process. Finally we give a detailed description of various approaches and algorithms used for grid service discovery.

3.1 Information services

Information services [26,39] are an integral part of the grid architecture. It is the foundation of how resources are defined and their state is known. More importantly, the user of the Grid gets a perspective of what a grid looks like, how it performs and what capabilities it has from information services.

Information services provide efficient access and manipulation of information about applications, resources and services in the Grid environment. In the context of grid, information refers to dynamic data or events used for status monitoring; relatively static data used for discovery; and any data that is logged. The users typically have little or no knowledge of the resources contributed by participants in the virtual organization [13,15]. This poses a significant obstacle to their use. For this reason, information services [8,42] designed to support the initial discovery and ongoing monitoring of the existence and characteristics of resources, services, computations, and other entities are a vital part of a Grid system.

Information services provide fundamental mechanisms for discovery and monitoring, and hence for planning and adapting application behavior. In grid systems, the discovery, characterization, and monitoring of resources, services, and computations are challenging problems due to the considerable diversity, large numbers, dynamic behavior; and geographical distribution of the entities in which a user might be interested.

An information service for a grid environment must be able to handle a wide range of queries and many different types of resources. Furthermore, resource status and dynamic changes in VO membership must be handled as well as dynamic addition and deletion of information sources.

Existing technologies do not address these requirements. Directory services [8,11] such as X.500 [28,45], LDAP, and UDDI [2,52] do not address explicitly the dynamic addition and deletion of information sources; in the case of X.500 and LDAP, there is also an assumption of a well-defined hierarchical organization, and current implementations tend not to support dynamic data well. Metadirectory services permit coupling multiple information sources, but otherwise inherit the quality of their database engine. Service discovery [36,57] services rely on multicast to scope requests, which is inappropriate when virtual and physical organizational structures do not correspond. Monitoring and event systems support information delivery, using disparate protocols, but not the scalable discovery of information sources.

3.2 Resource Discovery

Resources are the base components of a Grid environment. They form the low level entities that are accessed and used to fulfill a user request. Different resources can have the same functional capabilities but they may have different access policies associated, different time access, etc. The main motivation for grid computing is the sharing of these resources and the first thing needed for efficient sharing is a well-defined resource discovery process.

The grid resource discovery [25,30] process can be defined as the process of matching a query for resources, described in terms of required characteristics, to a set of resources that meet the expressed requirements. It takes as input a user request and returns a list of resources or services that can possibly fulfill the given request.

To make information available to users quickly and reliably, an effective and efficient resource discovery mechanism is crucial. However, grid resources are potentially very

large in number and variety; individual resources are not centrally controlled, and they can enter and leave the grid systems at any time. For these reasons, resource discovery in large-scale grids can be very challenging.

3.3 Steps in Resource Discovery Process

Most users perform resource discovery in the three steps [40] described below.

- authorization filtering,
- job requirement knowledge, and
- filtering to meet the minimal job requirements.

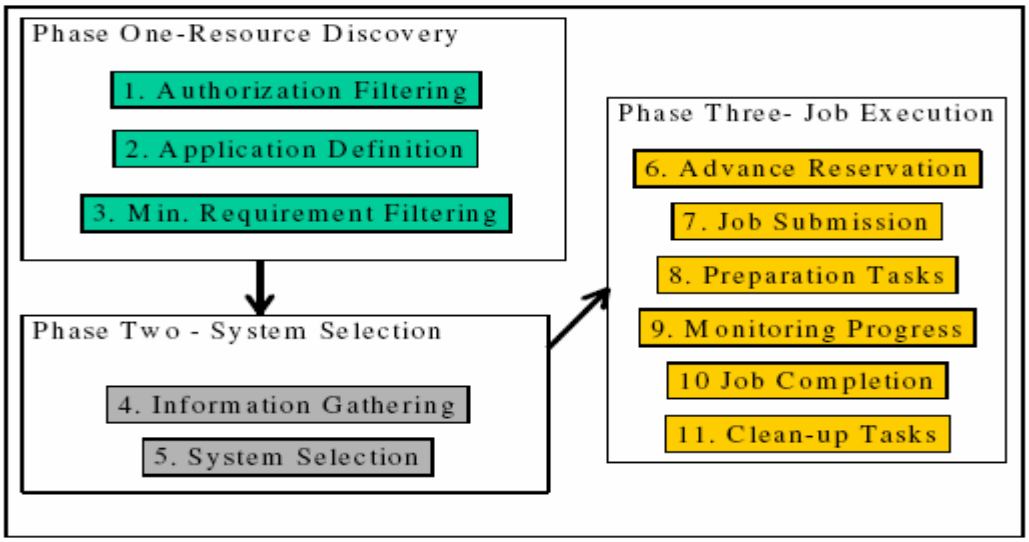

Figure 3.1 Resource Brokering Phases [41]

Step 1: Authorization Filtering

It is generally assumed that a user will know which resources he or she has access to in terms of basic services. At the end of this step the user will have a list of machines or resources to which he or she has access.

Step 2: Application requirement definition

In order to proceed in resource discovery, the user must be able to specify some minimal set of job requirements in order to further filter the set of feasible resources (in Step 3).

The set of possible job requirements can be very broad, and vary significantly between jobs. It may include static details, such as the operating system or hardware for which a binary of the code is available, or that the code is best suited to a specific architecture. Dynamic details are also possible, for example a minimum RAM requirement, connectivity needed, temporary space needed, etc. This may include any information about the job that should be specified to make sure that the job can be matched to a set of resources.

Step 3: Minimal requirement filtering

Given a set of resources to which a user has access and the minimal set of requirements the job has, the third step in the resource discovery phase is to filter out the resources that do not meet the minimal job requirements. The user generally does this step by going through the list of resources and eliminating the ones that do not meet the job requirements as much as they're known.

3.4 Resource Discovery Approaches

Approaches to resource discovery can be broadly classified as **Query Based** and **Agent Based**.

In a query based discovery the resource information store is queried for resource availability. Most contemporary grid systems follow this approach. In agent based discovery agents traverse the grid system to gather information about resource availability.

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 rely on an a fixed function query engine.

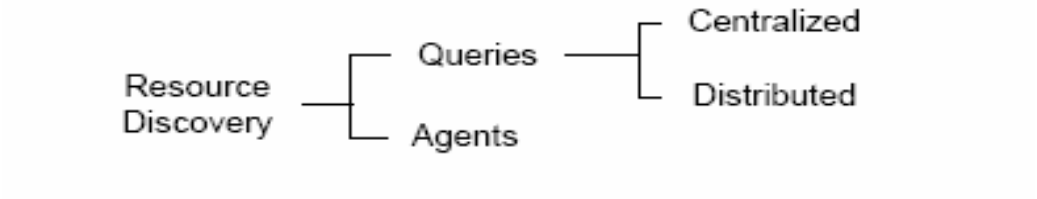

Figure 3.2 Resource Discovery Taxonomy [34]

3.4.1 Agent-Based Resource Discovery

The A4 (Agile Architecture and Autonomous Agents) methodology [1] can be used to build large-scale distributed software systems that exhibit highly dynamic behavior. An agent can represent one or more local high performance resources at the meta-level of the system. It is intended that an entire system be built of a hierarchy of identical agents with the same functionality. Agents are considered both service providers and service requestors.

A4 system is a distributed software system based on the idea of federating agents, which can provide services to a large-scale, dynamic, multi-agent system. The A4 model comprises a hierarchical model, a discovery model and a coordination model. The hierarchical model describes a simple method that can organize large numbers of agents. The discovery model solves the problem of coordinating the services of different agents, so that they locate each other, this being the basis of the hierarchical model. The coordination model focuses on the way that services can be organized to provide more complex services.

The hierarchical model is illustrated in Figure 3.3. The single type of component that composes the system is the agent. No agent has more special purposes or functions than any other. Every agent can act as a router between the request and the service. The broker is the agent that heads the whole hierarchy maintaining all the service information of the system. The coordinator is the agent that heads a sub-hierarchy. Only the leaf-node of the hierarchy is named as an agent.

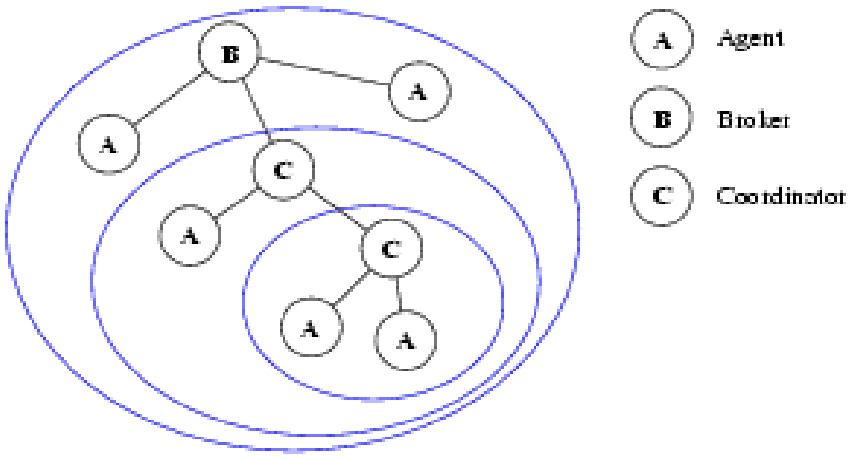

Figure 3.3 Hierarchical Model [1]

The name of this methodology i.e., A4 (Agile Architecture and Autonomous Agents) is attributed to the following characteristics of the system.

- While many agents can be utilized as a large-scale **multi-agent system**, the agents are not predetermined to work together. They possess their own motivation, resource and environment.

- **Autonomy** is used to describe the character of an agent, with regard to its intelligence and social ability. An agent can fulfill high-level tasks directly or through cooperation with other agents.

- **Architecture** is used to provide a framework for interaction between agents. For example, multiple agents can be organized into a hierarchy.

- **Agility** is used to describe the character of the architecture, and implies quick adaptation to environmental change. While autonomy provides the system with component-level adaptability, agility provides the architecture-level adaptability of the system.

The implementation of system functions is abstracted to the processes of service advertisement and service discovery.

- **Service Advertisement**

  The service information of an agent can be advertised either upwards or downwards within the hierarchy. There are different strategies to decide when and how to advertise a service with different performance.

- **Service Discovery**

  When an agent requires a service, it will check its own knowledge to find whether it has the related service information. If it has, it will contact the target agent, or it may contact the other agents in the hierarchy until the target service is reached.

Service advertisement and discovery can be abstracted into the look-up and maintainers of the Agent Capabilities Tables (ACT) in each agent. All the agents are able to change their services periodically. The identity of an agent may also change when it moves from one host to the other. It is these dynamics that increases the difficulty of service discovery. The most essential problem is how an agent advertises its services and coordinates with the other agents to find the required services in the most efficient way. The two extreme situations that exist are a data-pull model (no service advertisement) where complex service discovery is required, and data-push model (full service advertisement) where no service discovery is necessary.

Different systems can use different discovery models to achieve high performance. For example, static systems, where frequency of service information change is far less than the frequency of the request for the service, can use pure data-push model to achieve high performance service discovery. The DNS (Domain Name System) is a good example of this situation. Extreme dynamic systems, where the frequency of the service information change is far more than the frequency of the request, can use pure data-pull model to achieve high performance. Most practical systems require a middle point between these two.

There are four main performance metrics of the agent system: resource discovery speed, system efficiency (the balance between resource advertisement and discovery), load balancing and discovery success rate. These resource discovery metrics are often conflictive, in that not all metrics can be high at the same time. For example, a quick discovery speed does not equate to high efficiency, as it may be achieved through the high workload encountered in resource advertisement and data maintenance, leading to low system efficiency. It is necessary to find the critical factors of the practical system, and use the different performance optimization strategies to obtain high performance. These strategies include the use of cache, local and global knowledge, limited service lifetime, and limited scope for service advertisement and discovery.

3.4.2 Query-based Resource Discovery

Most of the contemporary grid systems use parameterized queries that are sent across the network to the nearest directory, which uses its query engine to execute the query against the database contents. Query based system are further characterized depending on whether the query is executed against a distributed database or a centralized database. There are a number of query-based approaches as described below.

3.4.2.1 Directory services

A directory service is a service that provides read-optimized access to general data about entities, such as people, corporations, and computers via a network protocol. Often, directory services [8, 51] include mechanisms for replication and data distribution.

One approach to constructing a Grid information service is to push all information into a directory service. While this approach pioneered information services [26,42] for the Grid, the strategy of collecting all information into a database inevitably limited scalability and reliability. X.500, LDAP and UDDI are some of the most popular directory service standards.

3.4.2.1.1 X.500

The X.500 standard defines a directory service [11,51] that can be used to provide extensible distributed directory services within a wide area environment. X.500 [28,45] provides a framework that could, in principle, be used to organize the information that is of interest to us. However, it is complex and requires ISO protocols and the heavyweight ASN.1 encodings of data. For these reasons, it is not widely used.

3.4.2.1.2 LDAP

The Lightweight Directory Access Protocol (LDAP) [35,48] is a simplified version of the X.500 Directory Access Protocol (DAP) which specifies a means of organizing and accessing information directories over the Internet. It removes the requirement for an ISO protocol stack, defining a standard wire protocol based on the IP protocol suite. It also simplifies the data encoding and command set of X.500 and defines a standard API for directory access.

LDAP is often used in organizations as a means of storing personnel, service, and network topology information. Users of LDAP [48,56] access information organized in a hierarchical directory tree. Each level of the tree contains attribute-value pairs of information as well as links to lower levels. While LDAP by itself is not a candidate for the role of resource discovery solutions in OGSA Grids (it is a means of storage and organization, not of description and discovery), LDAP's flexibility has made it the choice for a number of resource discovery solutions, including MDS [8,39], the Monitoring and Discovery System used by the Globus toolkit [10,19].

The **Globus Toolkit's Monitoring and Discovery System (MDS)** [11,39] uses LDAP to publish information about the current state of resources in a Grid environment. Information is structured as a set of entries, where each entry comprises zero or more attribute-value pairs. The type of an entry, called its object class, specifies mandatory and optional attributes. An Index Service provides the capabilities of the OGSA Information Service and includes data refreshing mechanisms that prevent stale data from being returned in query results. MDS's Trigger Service monitors MDS's data catalog for

certain preset conditions, providing a means for asynchronous alarms and warnings to be sent to interested parties.

3.4.2.1.3 UDDI

UDDI [2,52] is an OASIS standard protocol that defines a "standard method of publishing and discovering network-based software components in a Service Oriented Architecture (SOA)". It provides its functionality through four principle entities: the businessEntity, businessService, tModel, and the bindingTemplate.

The businessEntity is the largest container within UDDI. It contains information about any organizational unit which publishes services for consumption.

In the Grid context, businessEntities can be used to hierarchically separate and form relationships between different organizational groups within a Grid or multiple Grids.

Each service that a businessEntity offers is described by a businessService object. These objects provide information about the service name, categorization, and any other useful details added in a variety of attribute lists. The information is purely descriptive and does not include any instructions for accessing or using the service.

The bindingTemplate object represents the link between abstract businessService descriptions and actual endpoints at which these services may be accessed. Like all objects in UDDI, bindingTemplates are given universally unique identifiers, known as UUIDs, which are used as a key of reference. Each businessService object stores the UUID keys of bindingTemplates within the same businessEntity that provide instances of that service.

BindingTemplates may provide specialized information about a particular businessService endpoint through use of tModels. The tModel is the standard way to describe specific details about a particular businessService or bindingTemplate in UDDI. tModels contain lists of key-value pairs used for description and may be associated with

multiple objects in the UDDI database. They may also contain placeholders for URIs which point to descriptive information external to the UDDI registry.

However, to date, Web service-based Grids, such as those based on the emerging Open Grid Services Architecture (OGSA), have not utilized UDDI as a discovery mechanism due to the following reasons:

- It lacks a rich query model due to its lack of explicit data typing and its inability to easily perform bindingTemplate queries based on the values contained within associated tModels.
- It is not well equipped to handle environments that contain resource providers with unpredictable availability because of its limited support for the expiration of stale data.

3.4.2.2 Peer to Peer Approach

Ian Foster, Adriana Iamnitchi and Daniel C. Nurmi introduced the concept of peer to peer approach [31] for grid service discovery. According to them, a resource discovery solution should consist of the following four components:

- **Membership protocol**: The membership protocol specifies how new nodes join the network and how nodes learn about each other.

- **Overlay construction function**: The overlay construction function selects the set of collaborators from the local membership list. This set may be limited by such factors as available bandwidth, message-processing load, security or administrative policies, and topology specifications.

- **Preprocessing**: Preprocessing refers to off-line processing used to enhance search performance prior to executing requests. An example of a preprocessing technique is dissemination of resource descriptions [53], 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 function has a local and a remote component described below.
  - o Local processing: The local component looks up a request in the local information, processes aggregated requests (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.
  - o Remote processing: The remote component implements the request propagation rule i.e., sending the requests to other peers through various mechanisms such as flooding, forwarding, epidemic communication etc.

3.4.2.3 Parameter Based Approach

Muthucumaru Maheswaran and Klaus Krauter introduced a concept called the "Grid potential" [38] that encapsulates the relative processing capabilities of the different machines and networks that constitute the Grid.

The Grid potential concept is similar to the time-to-live idea used in the Internet. Informally, the Grid potential at a point in the Grid can be considered as the computing power that can be delivered to an application at that point on the Grid. The computing power that can be delivered to an application depends on the machines that are present in the vicinity and the networks that are used to interconnect them. Consequently, a high-performance machine when connected to the Grid will induce a large Grid potential. This potential, however, will decay as the launch point of the application moves away from the point at which the machine is connected to the Grid. The rate of potential decay depends on the network link capacities.

A node in the Grid has several attributes that can be categorized as rate-based attributes and non rate-based attributes. Examples of rate-based attributes include CPU speed, FLOP rating, sustained memory access rate, and sustained disk access rate. The Grid potential is based on the computing power or operating rate of a node. Therefore, to characterize a node for deriving the Grid potential only rate-based attributes are considered. A node (service) in the Grid can then be characterized by a vector where each

element of the vector is an attribute-value pair and discovery can be done using this vector.

3.4.2.4 Qos Based Approach

Yun Huang and Nalini Venkatasubramanian [29] addressed the problem of resource discovery in a grid based multimedia environment, where the resources providers, i.e. servers, are not available all the time but are only intermittently available.

Multimedia applications are resource intensive, and consume significant network, storage and computational resources. Furthermore, multimedia applications also have Quality-of-Service (QoS) requirements that specify the extent to which properties such as timeliness can be violated. Quality of Service (QoS) [17,18] is the summarizing term for the main non-functional properties of such systems which mainly includes reliability, security, performance and adaptability.

Effective load management in such environments requires a resource discovery mechanism that will ensure the continuity of data to the user while servers are intermittently available. Huang and Venkatasubramanian proposed a generalized resource discovery algorithm [29] based on graph-theoretic approach.

3.4.2.5 ClassAd Matchmaking

Matchmaking is a distributed resource management mechanism developed as part of the Condor project for Grid systems. The matchmaking is based on the idea that resources providing services and clients requesting service advertise their characteristics and requirements using classified advertisements (classads) [43,44]. A matchmaker service that may be either centralized or distributed matches the client requests to the appropriate resources. The matchmaking framework includes several components of a resource discovery mechanism.

The classad specification defines the syntax and semantic rules for specifying the evaluating the attributes associated with the characteristics and requirements. The

advertising protocol lays down the rules for disseminating the advertisements. The following figure shows the actions involved in the matchmaking process.

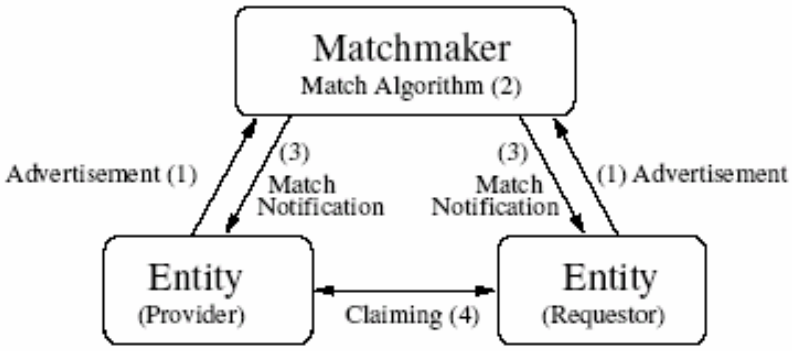

Figure 3.4 Actions involved in Matchmaking process [44]

3.4.2.6 Ontology based Semantic Matching

An ontology defines a common vocabulary that facilitates the sharing of information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and their relations. In the Grid environment (GE) where so many different implementations are available, the need for semantic matching based on a defined ontology becomes increasingly important. The loose coupling between resource and request descriptions remove the tight coordination requirement between resource providers and consumers.

A. Harth, H. Tangmunarunkit ,C.Kesselman and S. decker have designed and prototyped a matchmaker [26] using TRIPLE to use ontologies encoded in W3C's Resource Description Format (RDF) and rules (based on Horn logic and FLogic) for resource matching. Resource descriptions, request descriptions, and usage policies are all independently modeled and syntactically and semantically described using RDF schema. Finally, inference rules are used for reasoning about the characteristics of a request, available resources, and usage policies to appropriately find a resource that satisfies the request requirements.

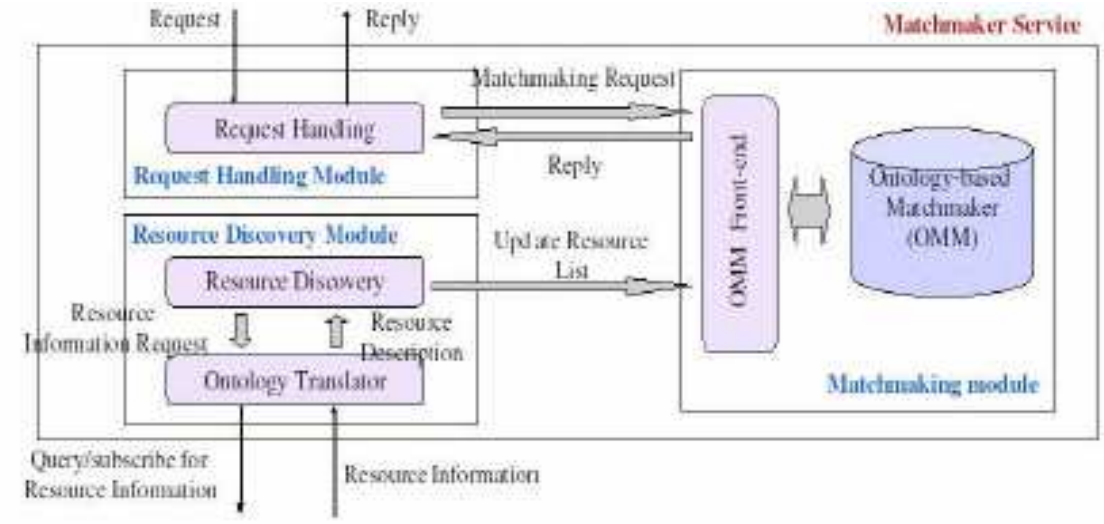

Figure 3.5 Architecture of the Ontology-based Resource Matchmaker Service [26]

3.5 Algorithms

This section discusses the various algorithms developed for discovering services in a grid environment.

3.5.1 Flooding Algorithm

In the Flooding algorithm [25], a machine is initially configured to have a fixed set of neighboring machines, and direct communication is only allowed with machines in this set. In terms of the graph, a node only communicates over the edges that were initially in the graph; new edges that are added to the graph are not used for communication. The edges that constitute the "initial neighbors" are not necessarily the links in the underlying physical network, but rather they are virtual links, each possibly corresponding to a path in the underlying network.

3.5.2 Swamping algorithm

The Swamping algorithm [25] is identical to the Flooding algorithm except that machines may now open connections with all their current neighbors, not just their initial neighbors. Also since the neighbor sets change, all of the current neighbor set is transferred, not just the updates.

The advantage of the Swamping algorithm is that the graph always converges to a complete graph in O(logn) steps, irrespective of the initial configuration. The disadvantage of the Swamping algorithm is that the network communication complexity is increased.

The bottom line is that the Swamping algorithm is very fast (only O(log n) rounds), but this speed is obtained at the cost of wasted communication where many machines are being told of machines they already know about.

3.5.3 The Random Pointer Jump Algorithm

The disadvantage of the Swamping algorithm is that the communication complexity grows quickly. To reduce the communication complexity one might consider having each machine communicate with only one randomly chosen neighbor during each round.

The Random Pointer Jump Algorithm [25] works as follows:
In each round, each machine ‘a’ contacts a random neighbor ‘b’ in its set of machines. The chosen neighbor ‘b’ then sends its own set of neighboring machines to machine ‘a’, which then merges the two sets i.e., its own set and its neighbor’s set.

The Random Pointer Jump Algorithm can only be applied to strongly connected networks (i.e., there must exist a path between every pair of machines), because otherwise the graph will never converge to a complete graph. Consider for example the graph with two nodes and a single directed edge between them: the remaining edge cannot be formed.

3. 6 Conclusion

This chapter described information services, resource discovery process and steps followed by this process, various approaches and algorithms that can be used for grid service discovery.

In the next chapter we will focus on the problems encountered in the existing approaches and possible solutions to these problems.

Chapter 4

Motivation for Semantic Based Approach

This chapter gives a detailed description of the problem found in existing approaches of service discovery in a grid environment and the various semantic solutions that can be used to solve this problem.

4.1 Problem with Existing Approaches

The description of a service is essential for automated discovery and search, selection, matching, composition and interoperation, invocation and execution monitoring; and different middleware specify different rules for describing a service. Hence the information gathered from these diverse sources tends to be semantically heterogeneous and needs to be correlated. The sources of information are equally different or heterogeneous. None of the approaches described in the previous chapter is capable of correlating this information. Thus there is a need to construct systems that are capable of answering intelligently by understanding the terms involved and the relationships between them.

Existing resource description and resource selection in the Grid is highly constrained. Traditional service discovery is done based upon symmetric attribute-based keyword matching with the support of simple query languages. In these systems, the values of attributes advertised by nodes are compared with those required by jobs. For the comparison to be meaningful and effective, the node providers and consumers have to agree upon attribute names and values beforehand. However, exact keyword matching and coordination between providers and consumers make such system inflexible and difficult to extend to new characteristics or concepts. Moreover, in a heterogeneous multi-institutional environment such as the Grid, it is difficult to enforce the syntax of node descriptions since a grid is setup using a number of middleware, all of which describe the services in their own way. Thus there is a need for a system that is independent of the syntax used for resource descriptions and provides a semantic

approach for finding services based on their concept similarity rather than exact matching of their attribute value.

In the real world individuals want to express the features of the desired services in the most effective manner using a natural language like English. Thus, we need semantic interoperability to go beyond simple keyword matching of attribute terms and comprehend the deeper meaning. Bridging the semantic gap between heterogeneous sources to answer end user queries is a prerequisite and key challenge to support global information systems.

In the existing approaches, an advertisement and a request are considered to be "sufficiently similar" when they describe exactly the same service. This definition is too restrictive, because providers and requesters have no prior agreement on how a service is represented and additionally, they have very different objectives. A restrictive criterion on matchmaking is therefore bound to fail to recognize similarities between advertisements and requests.

It is necessary to allow matchmaking engines to perform flexible matches, those that recognize the degree of similarity between advertisements and requests in order to provide a softer definition of "sufficiently similar". Service requesters should be allowed to decide the degree of flexibility that they grant to the system. By increasing the flexibility of a match, they increase the likelihood of finding services that match their requirements by reducing false negatives.

Matching can be categorized as:

- **Full match (request.input <= advertisement.output)**
  This is the situation where the node advertised contains more than or equal to the node requested. Full match is the best situation because the result returned might probably give more than what the requester expects.

- **Partial match (request.input > advertisement.output)** – This is the situation where the provided node cannot completely fulfill the request. In other words, the set of the advertised functions is a subset of the set of the requested functions.

- **Outer match (request.input and advertisement.output are disjoint):** No subsumption relation can be found between advertisement and request is identified.

4.2 Semantic Matching

All the existing service discovery approaches lack the ability of inexact matching. Semantic matching [4,6] is one of the approaches that provide the desired flexibility in matching. In keyword matching a demand is matched only to a perfectly matching supply, however semantic matching deals with concepts and logical relationships thus finding all those supplies that can fulfill a demand even to certain extent.

Semantic matching [21,46] is defined as a matching process that uses knowledge of meaning to broaden recall. It may use knowledge of synonyms, knowledge of part/whole relations and knowledge of class/subclass relationships.

4.3 Semantic Based Search Engine

The task of a semantic matching engine is to use its semantic understanding of the advertisement and request to recognize the degree of match and search for the services having similar concepts and not just the services described using the same keywords.

Some of the techniques that can be used for semantic matching are:

(a) **Dictionary and thesauri**: match words, phrases or parts of speech with a static or periodically maintained dictionary and thesaurus. Dictionaries such as WordNet can be used to identify and match terms in different directions, finding words that mean the same, are more general, or more specific.

(b) **Document analysis**: look for patterns through statistical space vector techniques, co-occurrences and application of pre-defined rules to find interesting patterns, within and across documents.

(c) **Ontology**: Ontology [50] is a specification of a conceptualization. In this context, specification refers to an explicit representation by some syntactic means. In contrast to schema languages (like XML Schema or DTDs) ontologies try to capture the semantics of a domain by deploying knowledge representation primitives, enabling a machine to (partially) understand the relationships between concepts in a domain. Additional knowledge can be captured by axioms or rules.

In this thesis work, we have implemented a semantic search engine which uses a thesaurus to discover services providing similar functionality. The implementation details and experimental results are described in the next chapter.

4.4 Conclusion

This chapter described the problem found in existing service discovery approaches and the possible semantic solutions to this problem.

In the next chapter we describe the implementation details and results which include setting up of the grid environment using various middleware in the Centre of Excellence for Grid Computing and implementation of semantic search engine which uses the thesaurus to find services offering capabilities similar to the one desired by the user.

Chapter 5

Implementation Details and Experimental Results

This chapter describes the implementation details including setting up of the grid environment using various middleware, installation of GridBus Broker and implementation of semantic search engine which uses thesaurus to find services offering capabilities similar to the one desired by the user.

5.1 Implementation Details

5.1.1 Setting up Grid Environment

A Grid environment was setup in the Centre of Excellence for Grid Computing by installing different middleware on different machines. Details of these middlewares are:

1. N1 grid engine 6 [49] was installed on three systems that included one master cum execution host and two execution hosts. The step-by-step installation procedure is given in appendix A.
2. Globus (Globus Toolkit 4.0) was also installed on three systems having linux operating system. The step-by-step installation procedure is given in appendix B.
3. Windows based version of Condor 6.6.10 was also installed on two systems. The procedure is given in appendix C.
4. Alchemi 1.03 was installed on a number of systems within the Centre of Excellence for Grid Computing and DSP lab. Its installation procedure is given in appendix D.

5.1.2 Installing GridBus Broker

As mentioned above, four different middleware were used to setup the grid environment in the Centre of Excellence for Grid computing. Since these middleware are based on different standards and are not interoperable with each other, we needed a common platform through which these middleware could communicate. GridBus broker [5,22] is

compatible with all the above mentioned middleware, hence we installed GridBus Broker by following the procedure given in this section.

1. Requirements

At Broker side (i.e. on the machine running the broker)

- Java Virtual Machine 1.4 or higher
- Valid grid certificates properly set up (if using remote Globus nodes)
  By default the certificates are places in the <USER_HOME>/.globus directory where <USER_HOME> is the user's home directory.
  For a user "belle" on a UNIX machine this would be: /home/belle/.globus
  For a user "belle" on a Windows NT/2000/XP machine this would be C:\Documents and Settings\belle\.globus.
- Additionally, some ports on the local node should be configured to be open so that the jobs can connect back to the broker. Please refer to the Globus documentation for more details.

At Remote Grid node side

For a compute resource:

• Middleware installation which is one of:

- Globus 2.4
- Globus 3.2
- Globus 4.0
- Alchemi 1.0 (Cross-platform manager)
- SGE Sun N1 Grid Engine 6
- Condor 6.6.10
- Unicore Gateway 4.1
- Open PBS 2.3

2. Installation process

Installing the broker is a simple process. The broker is distributed as a .tar.gz (and a .zip) archive that can be downloaded from

- http://www.gridbus.org/broker/2.4/gridbusbroker2.4.tar.gz.
- http://www.gridbus.org/broker/2.4/gridbusbroker2.4.zip.

The installation just involves unzipping the files to any directory and optionally setting the PATH environment variable to include the broker executable script (gbb.sh or gbb.bat depending on your OS).

Following are the steps to be followed:

- Unzip the archive to the directory where you want to install the broker. In case of Windows, you can use WinZip (if you download the .zip file) or WinRar (for the .tar.gz)

  In case of *nix, run the command:

  $ tar –zxvf gridbusbroker.2.4.tar.gz
- Set the GBB_HOME variable to the directory where you have installed the broker.
- Additionally, it is recommended to have the directory gridbus-broker2.4/bin added to the system PATH variable.

  For example, for a Bash shell:

  $ export PATH=$PATH:<broker-install-directory>/bin
- Set the permissions for the gbb.sh executable:

  $ chmod 755 gbb.sh
- Test the installation by running the broker from a shell:

  $ ./gbb.sh –test

The following message confirms that the configuration is ok:
Congratulations! You have successfully installed the Gridbus broker on your machine.

Figure 5.1 on the next page gives the snapshot of successful installation of GridBus Broker.

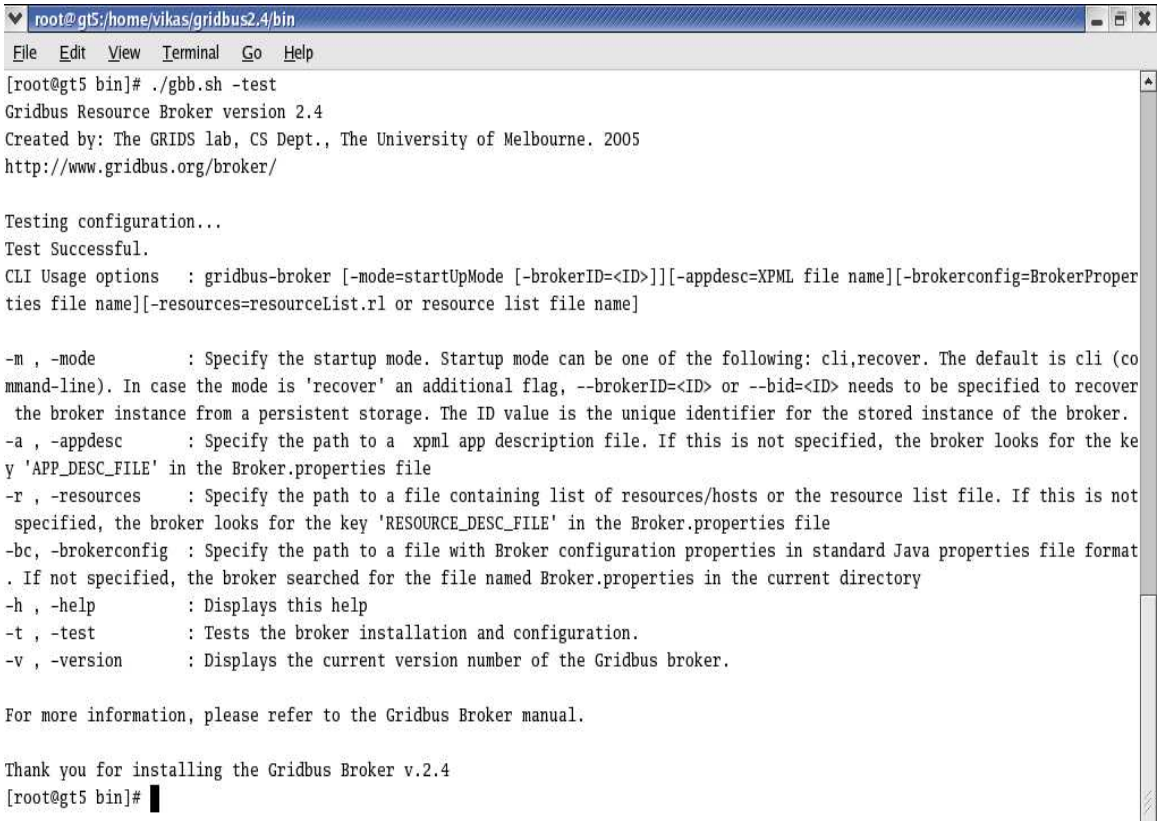

Figure 5.1 Successful installation of GridBus Broker v2.4

5.1.3 Grid Service Semantic Search Engine

The semantic search approach in grid environment exploits two novel technologies—the Grid and Semantic Web technologies—to provide a persistent online service for solving resource matching problem in the Grid.

GridBus Broker provides interoperability among different middleware but there is no support for semantics in the broker. It has a directory structure called Grid Market Directory (GMD) for discovery of services, which uses a keyword-based serial search technique to locate a service or resource. So we have developed a thesaurus-based search technique that can be integrated with GMD for discovering grid services semantically.

The schema of the designed database is based upon the schema of Grid Market Directory (GMD) [24,58] according to which the following information is stored about resources in the grid.

- Resource/Service Name
- Resource/Service Provider
- Location of the Resource/Service
- Cost of using hardware (in Rs. per hour)
- Cost of using software (in Rs. per hour)

A service needs to be registered by the service provider before it can be offered as a grid service. The details of service registration procedure are not considered in this thesis and can be taken up as a separate work.

The search engine has the following components:

1. Indexer

The indexer module extracts the service name whenever a service is registered by the service provider on the grid and records the URL where each service is located. The result is a lookup table that can provide all the URLs that point to the services with similar functionality. The user query is matched against this index.

2. User Interface

The user interface of the semantic search engine is just like the interface of a keyword based search engine, however semantic search engine enables the user to specify the query in a natural language like English. Thus the user can easily query for the desired service and find it even if he/she has no prior knowledge about the keywords that are used to describe the service by the service provider at the time of registration.

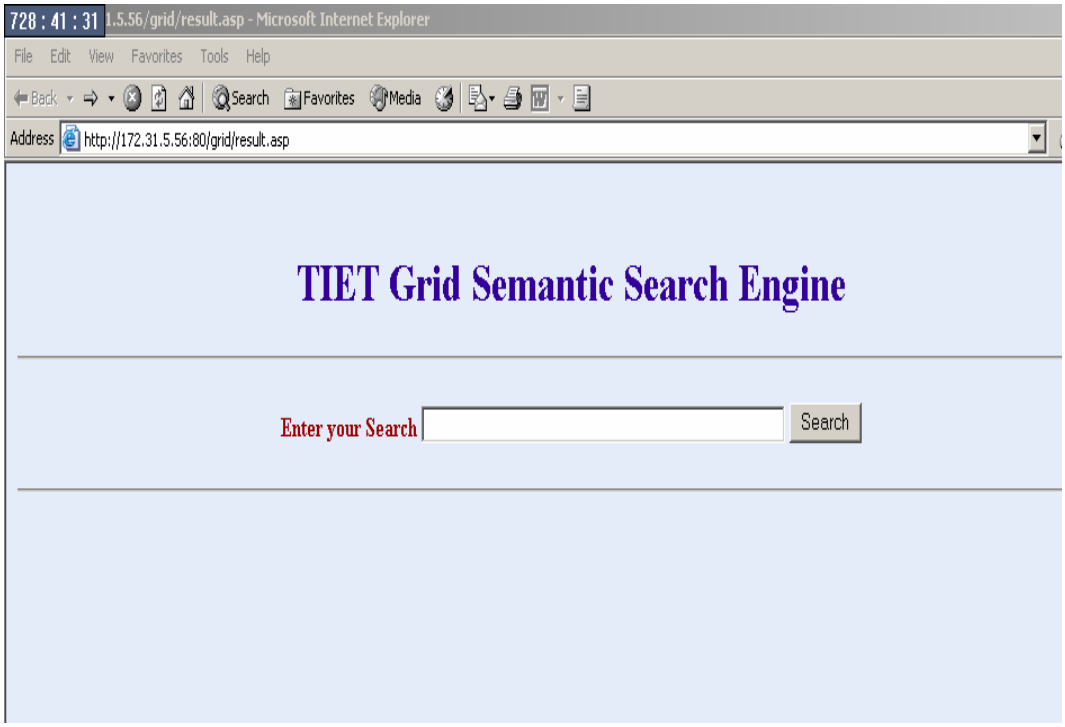

Figure 5.2 User Interface

3. Word Tokenization

As soon as a user inputs a query, the search engine tokenizes the query stream, i.e., break it down into understandable segments. Usually a token is defined as an alpha-numeric string that occurs between white space and/or punctuation. To tokenize words, we find word boundaries in a multi-term query using changes in font, presence of delimiters, etc.

4. Parsing

Users may employ special operators in their query, like Boolean logic, thus the system needs to parse the query into query terms and operators.

5. Creating the Query

The system recognizes single terms, phrases, and Named Entities. If any Boolean operator is used, it will also recognize the logical operators and create a representation containing logical sets of the terms to be AND'd, OR'd, or NOT'd.

6. Query Expansion

The query is expanded into all possible synonymous terms using WordNet thesaurus. The rich semantic information contained in the thesaurus identifies many other words that are semantically similar to the given input words.

7. Searching Technique

The expanded query is matched serially against the index created by indexer and the details of the records with matched keywords are fetched and displayed.

The flow of information between these components is shown in figure 5.2.

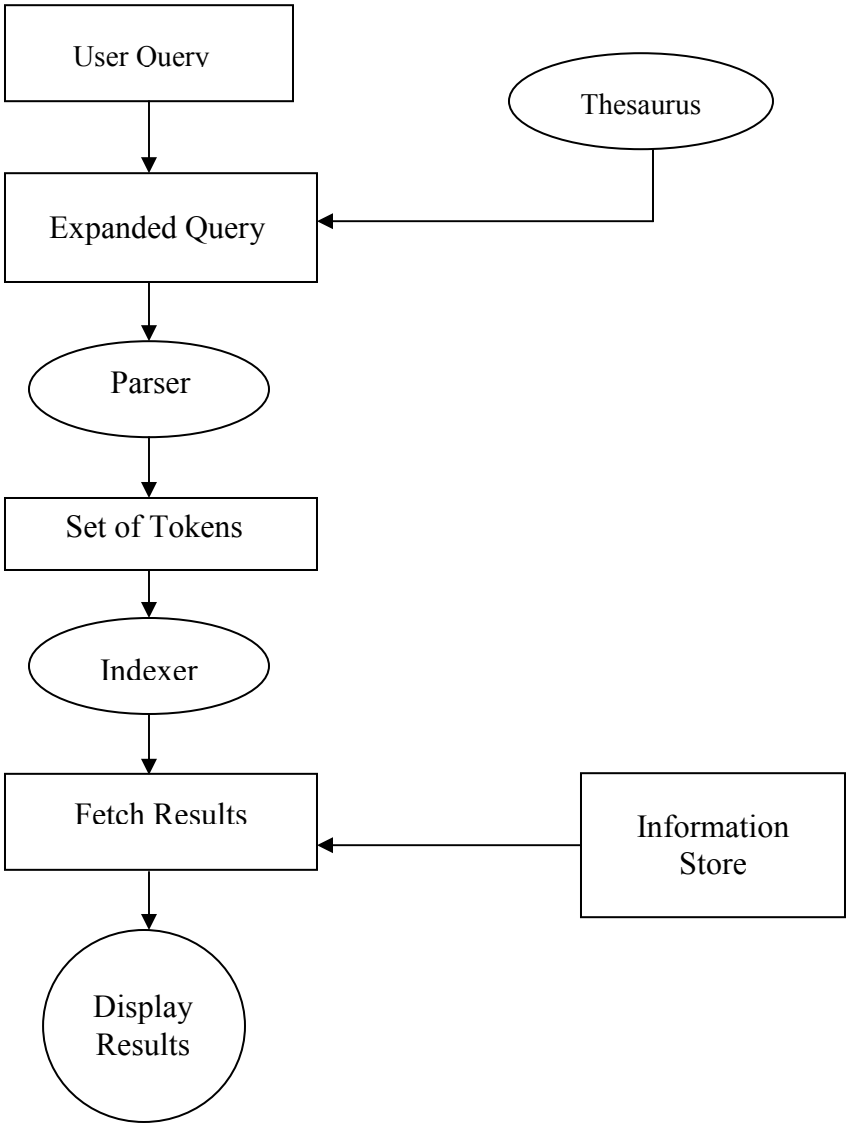

Figure 5.3 Flow of information between the components

5.2 Experimental Results

In this section we show the experimental results using some examples.

5.2.1 Service Discovery

Here we give two examples for searching services semantically. The service information present in the information store at the time of experiment is given below.

Service Name	Provider	Location	Hardware Cost (Rs per hour)	Software Cost (Rs per hour)
Extraterrestrial Intelligence	C-DAC,Pune	www.cdacp.ac.in	275	250
Interplanetary Signals	IIT,Bombay	www.iitb.ac.in	280	250
Weather Forecast	IIT,Delhi	www.iitd.ac.in	200	175
Weather Prediction	C-DAC,Noida	www.cdacn.ac.in	175	150
Weather Forecasting	IIT,Bombay	www.iitb.ac.in	180	175

Example 1

The user inputs a query to find a service that will interpret the signals received from planets outside the Earth. As shown above we have two services named “Extraterrestrial Intelligence” and “Interplanetary Signals” registered by different providers in the grid. Both these services provide similar functionality. The user queries “extraterrestrial signals”, “interplanetary signals”, “outside earth signals”, “signals from outside”, and “existence of life outside earth”, all convey the same meaning. These queries would have not given correct result if we used keyword search, however semantic search gives all the relevant results as shown in the screen shots given in the following pages.

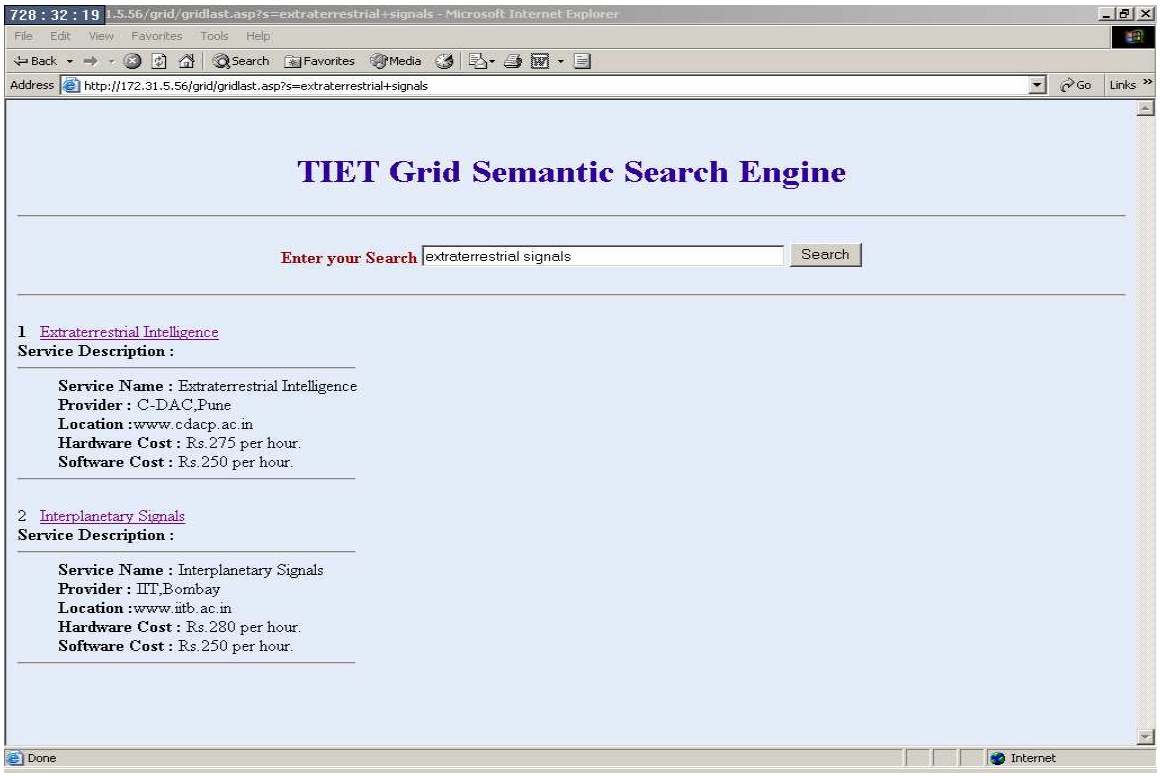

Figure 5.4 Search results for Query “extraterrestrial signals”

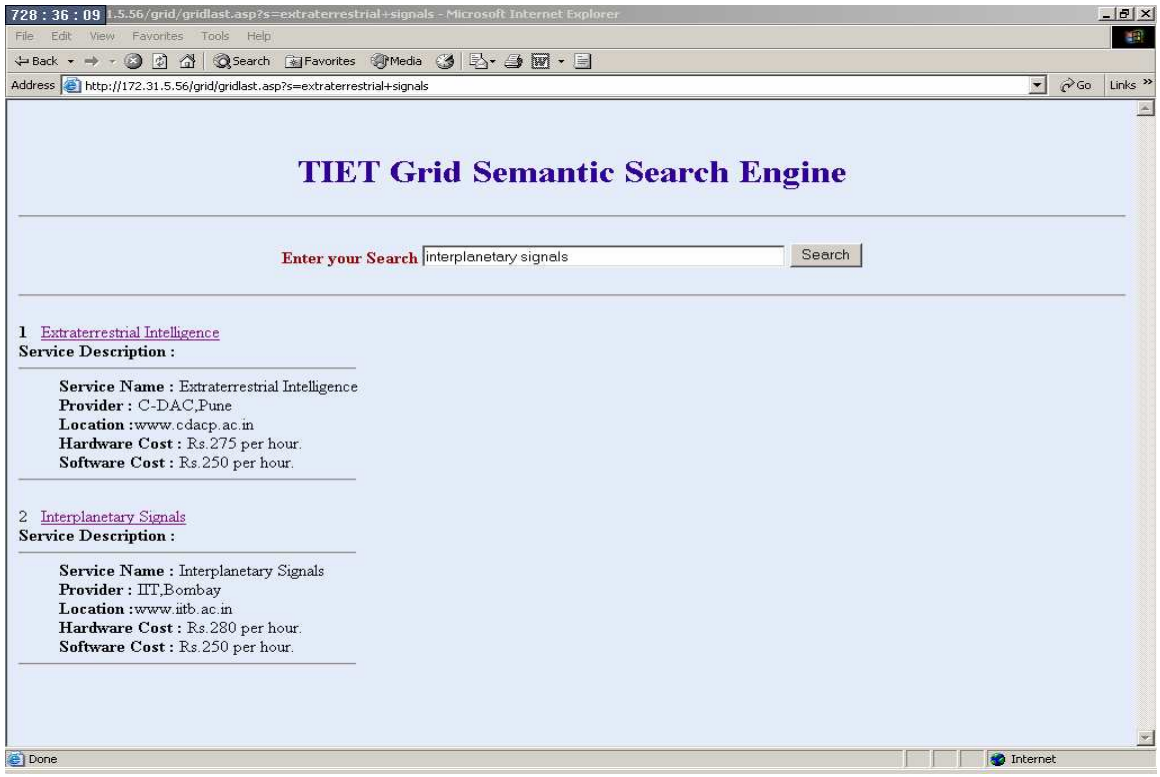

Figure 5.5 Search results for Query “interplanetary signals”

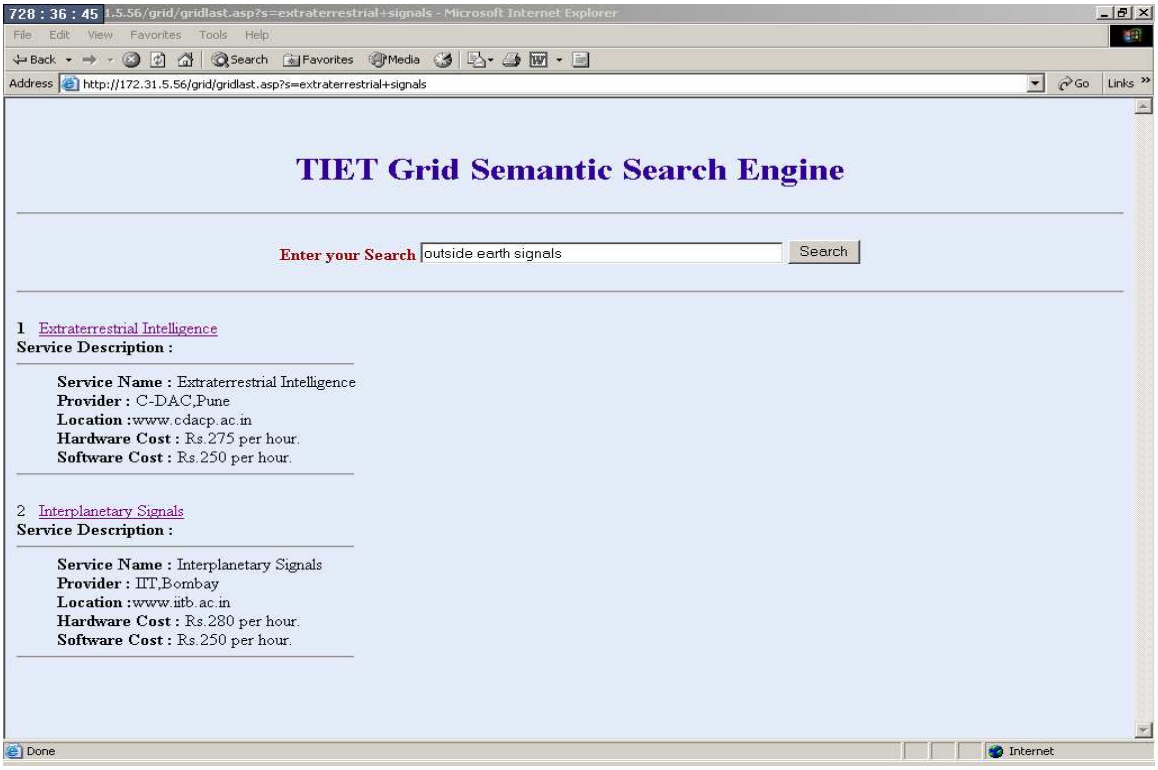

Figure 5.6 Search results for Query "outside earth signals"

Figure 5.7 Search results for Query "signals from outside"

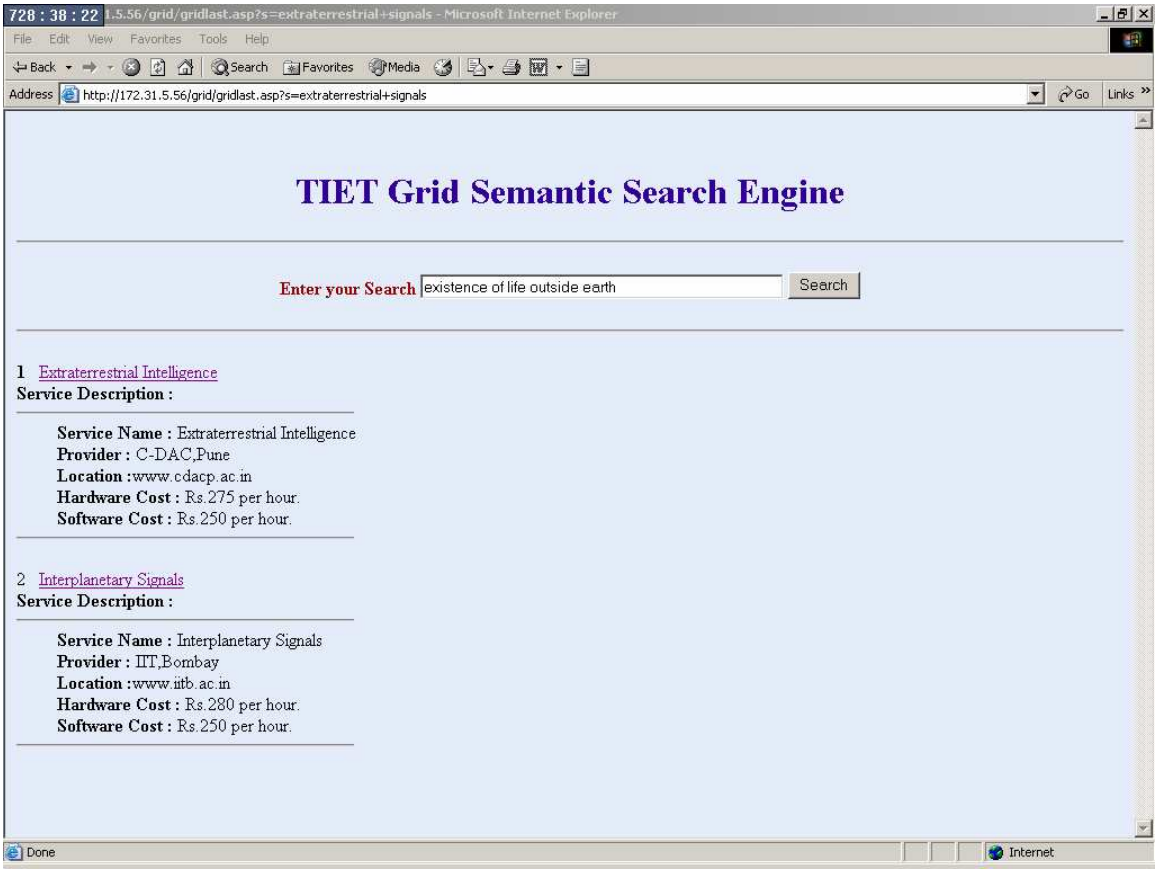

Figure 5.8 Search results for Query "existence of life outside earth"

Example 2

The user inputs a query to find a service that will predict the weather details. In our example we have three services named "Weather Forecast", "Weather Prediction" and "Weather Forecasting" registered by three different providers in the grid. The user can query the search engine in his/her own words without having prior knowledge to the exact names of these services and the result will include all these three services having similar functionality as shown in the screen shots given in the following pages.

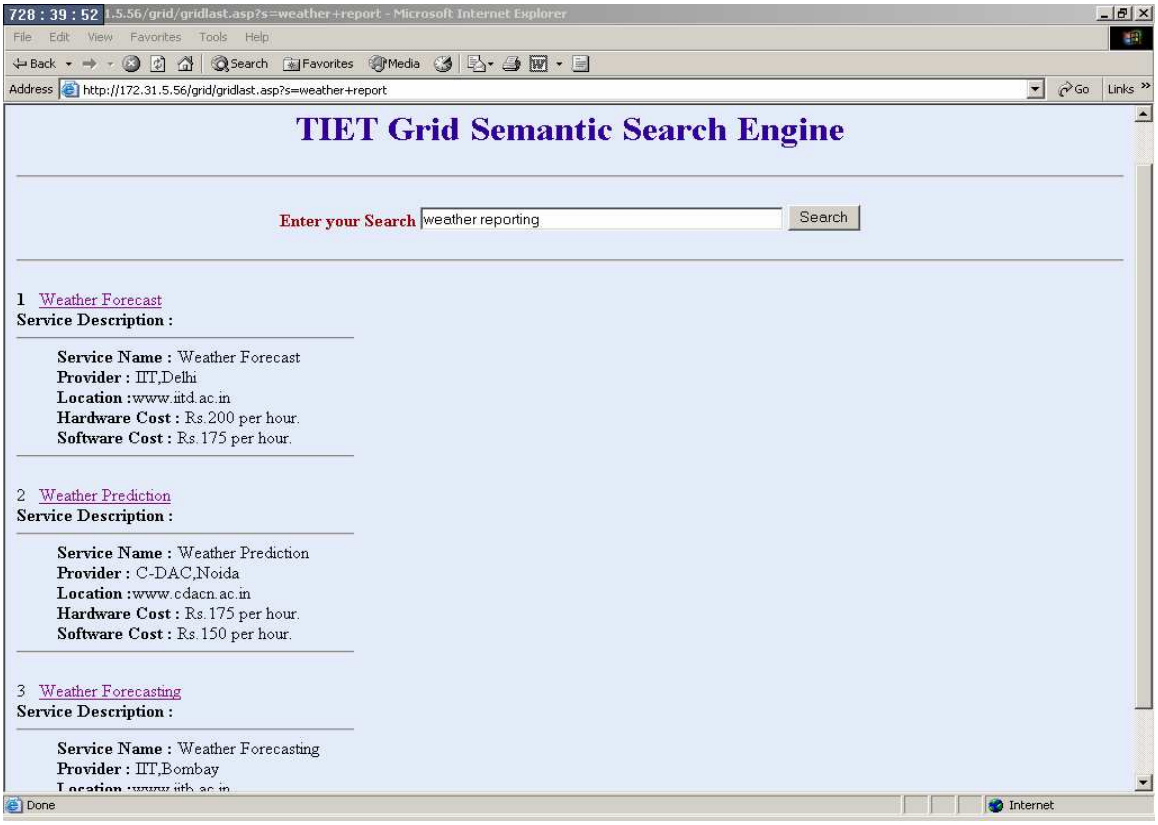

Figure 5.9 Search results for Query "weather reporting"

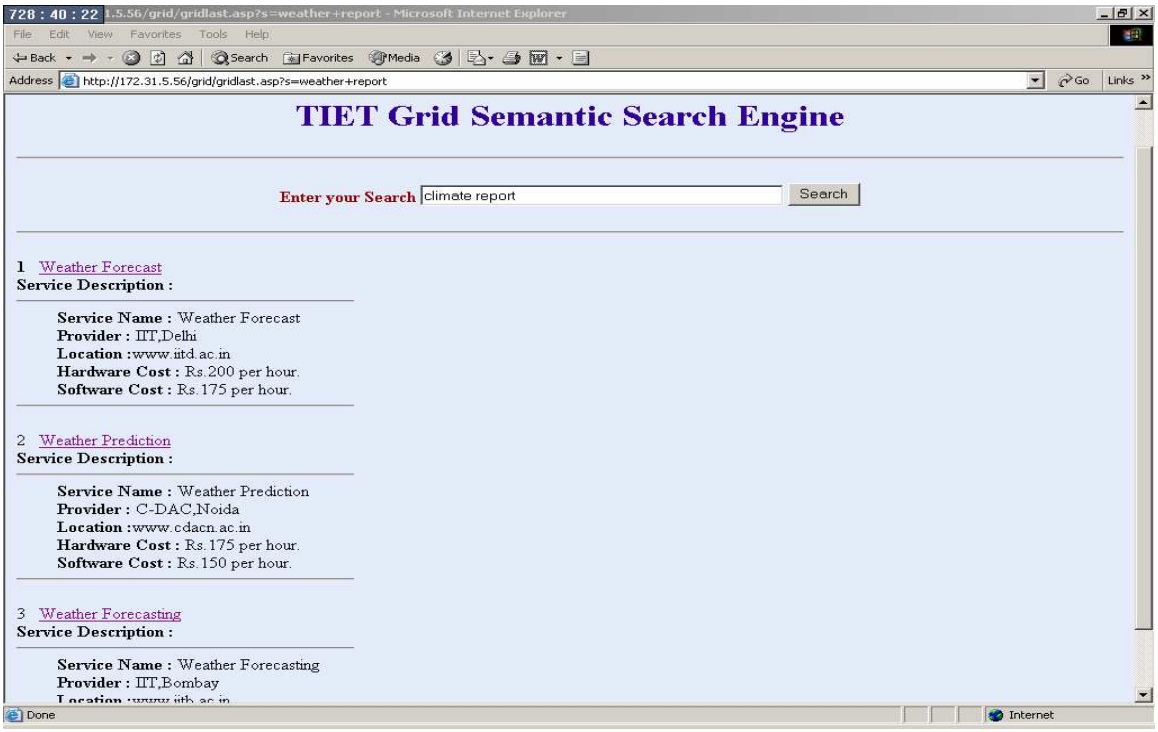

Figure 5.10 Search results for Query "climate report"

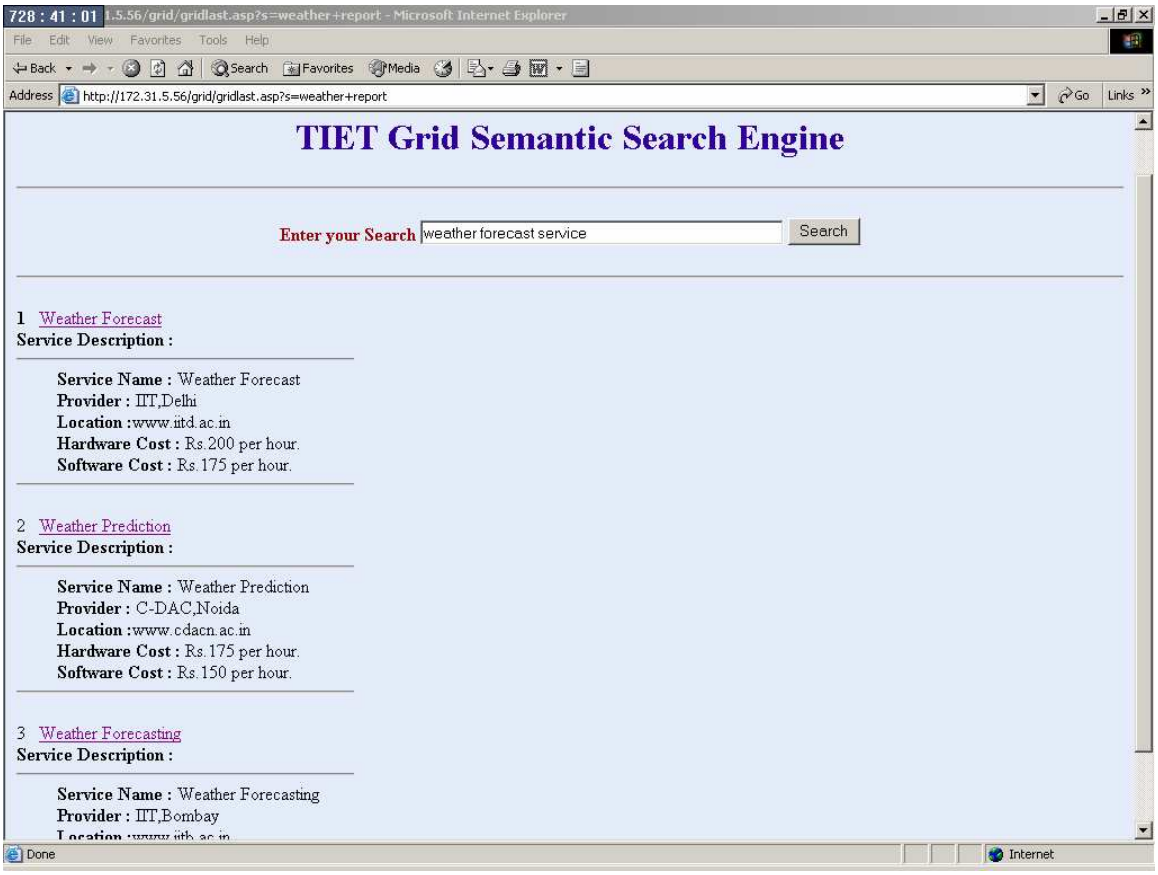

Figure 5.11 Search results for Query "weather forecast service"

Thus the Semantic Search Engine semantically searches the grid and displays all the services performing similar functionality although the service providers register them with different names at the time of registration.

5.2.2 Resource Discovery

We take three examples to illustrate semantic searching on desired resources. The user can specify his/her job requirements in terms of storage, speed or/and operating system. The resource information present in the information store at the time of experiment is given on the next page.

Resource Name	Provider	Location	H/w Cost (Rs per hour)	S/w Cost (Rs per hour)	Hard Disk (GB)	Processor Speed (GHz)	OS
TIET1	TIET, Patiala	www.tiet.ac.in	150	120	8	1.5	Red Hat Linux
TIET2	TIET, Patiala	www.tiet.ac.in	140	140	4	3.0	Solaris 10
TIET3	TIET, Patiala	www.tiet.ac.in	130	125	2	2.8	Unix
TIET4	TIET, Patiala	www.tiet.ac.in	200	180	8	3.0	Windows 2000

Example 1

In this example the user inputs a query for a machine that has processor speed of atleast 2.6 GHz. According to the configuration already mentioned, TIET2, TIET3, TIET4 satisy this request. The screenshot for the same is given below.

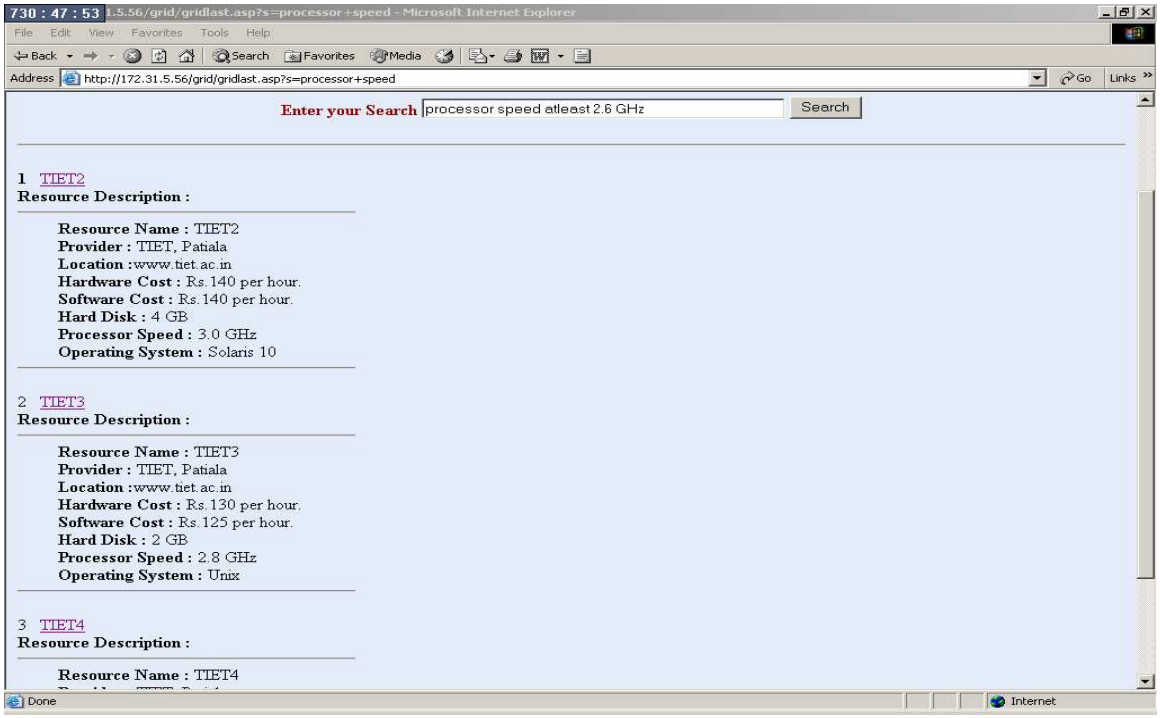

Figure 5.12 Search results for Query "processor speed atleast 2.6 GHz"

Example 2

In this example, user needs a machine with linux or unix operating system and processor speed atleast 1.5 GHz. Both TIET1 (Red Hat Linux, 1.5 GHz) and TIET3 (Unix, 2.8 GHz) satisfy this requirement.

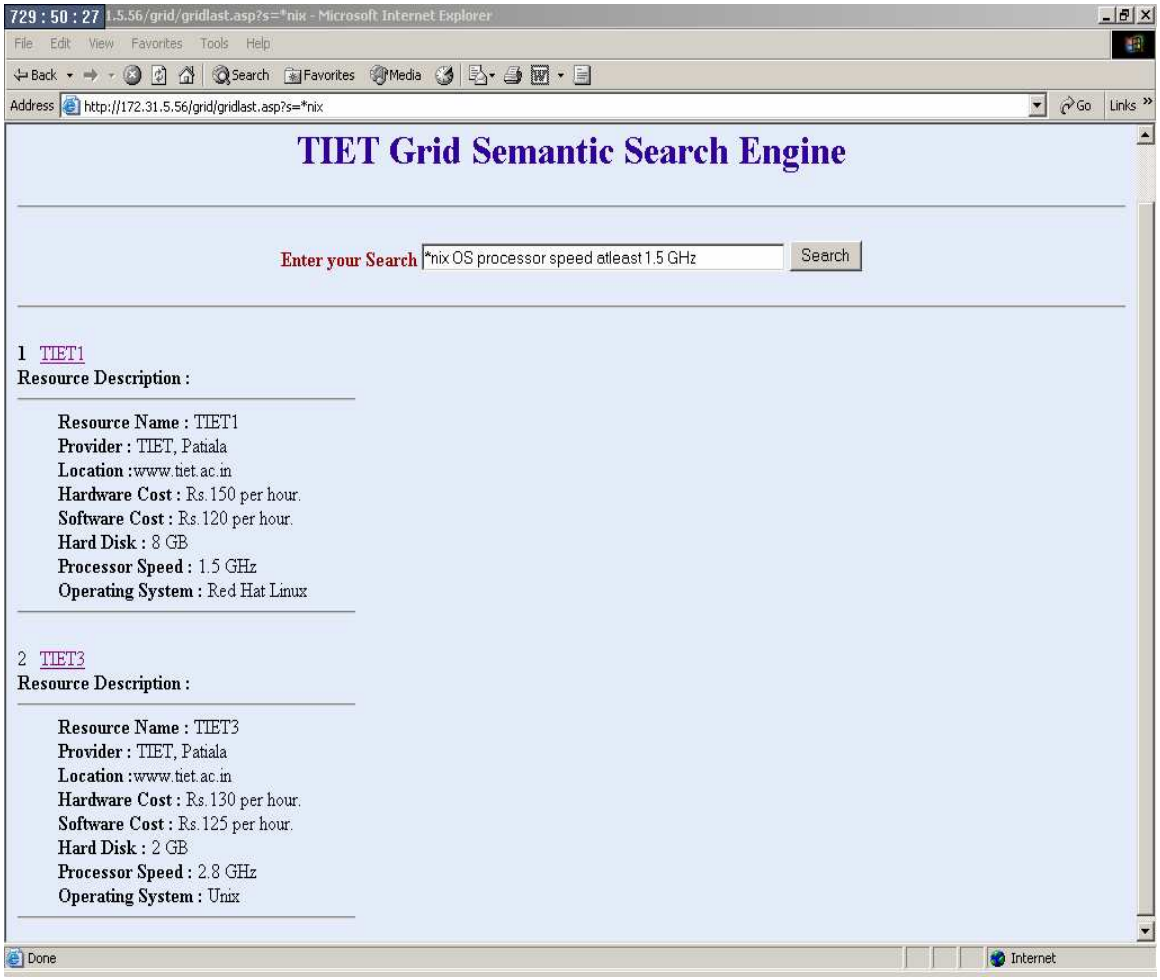

Figure 5.13 Search results for Query "*nix Os processor speed atleast 1.5 GHz"

Example 3

In this example the user requests for a machine with windows operating system having storage capacity of atleast 2 GB. He uses different words for storage capacity i.e., first time he queries for primary storage and next time for hard disk, in both the cases the result is machine TIET4 as shown below.

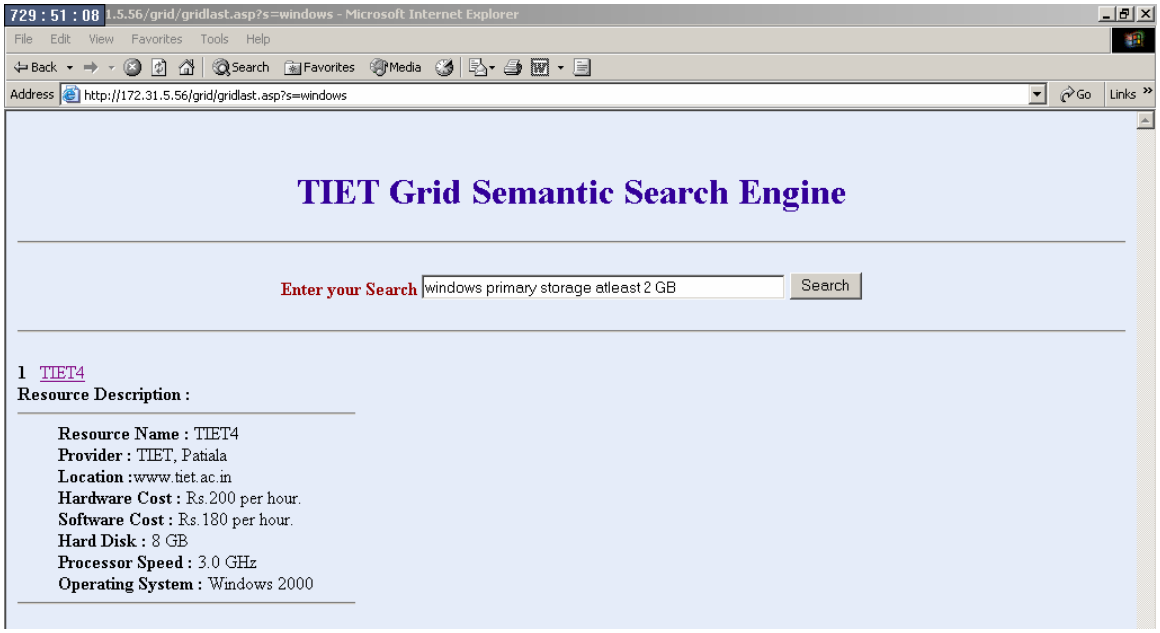

Figure 5.14 Search results for Query "windows primary storage atleast 2 GB"

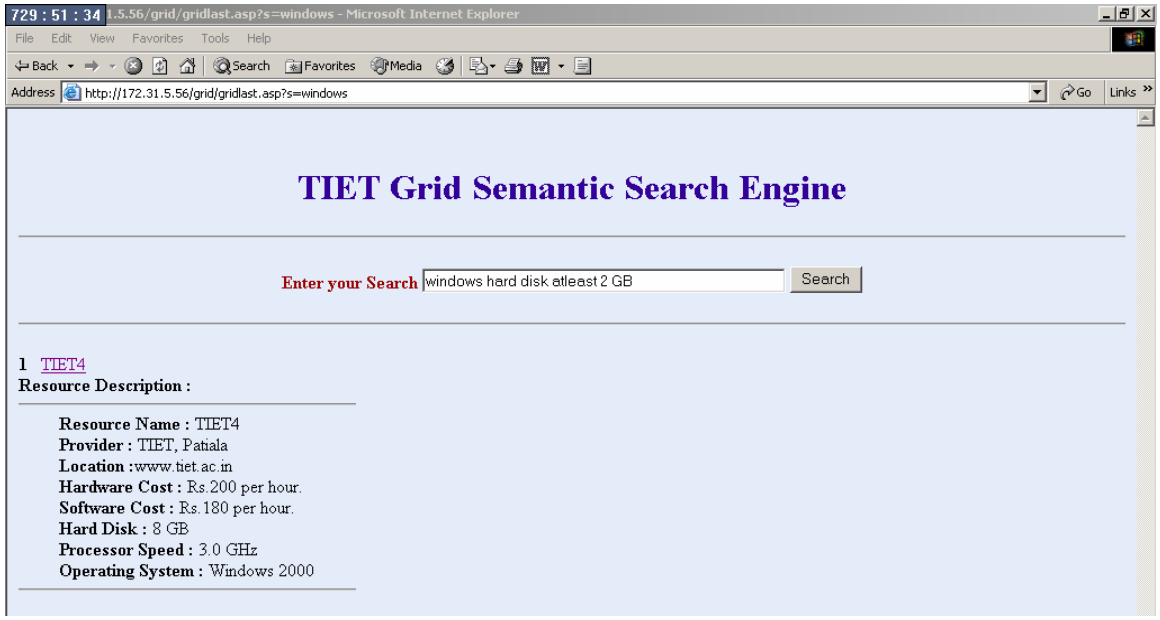

Figure 5.15 Search results for Query "windows hard disk atleast 2 GB"

5.3 Conclusion

This chapter focused on the setting up of grid environment, installation of GridBus Broker for discovery of services and implementation of Grid Service Semantic Search engine that displays the services/resources having similar attribute value and not just exact keyword matches.

The next chapter summarizes this thesis work and suggests features that can be incorporated in future for an enhanced semantic search.

Chapter 6

Conclusions and Future Scope of Work

This thesis work provided an insight into grid computing, the various approaches and algorithms currently used for service discovery in the grid environment, and the problems found in these approaches. Then it described the setting up of grid environment using different middlewares namely N1 Sun Grid Engine 6.0, Globus Toolkit 4.0, Condor 6.6.10 and Alchemi 1.03 in the Centre of Excellence for Grid Computing, installation of GridBus Broker and finally implementation of a semantic based approach for service discovery as a solution to the problems in traditional approaches.

6.1 Conclusions

Discovering a service that satisfies a user request sufficiently is a major issue in the grid environment due to its heterogeneous, dynamic and distributed nature. However, there is, as yet, no common standard for describing Grid resources. Since each middleware describes a service in its own way, information gathered from diverse sources tends to be semantically heterogeneous. Correlation of this kind of information using a semantics based approach is the topic of this thesis.

Approaches to resource discovery are broadly classified as query based and agent based. 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 rely on an a fixed function query engine.

Most of the popular middleware such as Globus, Condor etc use query based approaches where service discovery is done based upon symmetric attribute-based keyword matching. These systems lack the ability of inexact matching. In the real world individuals want to express the features of the desired services in the most effective manner using a natural language like English. Thus, we need semantic interoperability to

go beyond simple keyword matching of attribute terms and comprehend the deeper meaning.

In this thesis, we have attempted to achieve semantic interoperability by creating an index of synonyms of the attribute value of services. In particular, the semantic based approach is better than the resource discovery approaches used today in the following ways:

- Semantic search engine provides a flexible approach for discovery as it is not dependent on the syntax used for resource description by the different middleware.
- It provides a degree of freedom to the user whereby a user can input the query in natural language like English without having any prior knowledge about the exact names of the available services.
- It gives better results by finding services that have similar concepts not just same descriptions. Thus it returns all the relevant services and not just the services with matching keywords.

6.2 Future Scope of work

The semantic based search engine for grid service discovery described in this thesis can be further improved by incorporating a knowledge base that will automatically update the the collection of similar services .

Also, in future a full-fledged integrated user interface (portal) can be developed to manage all aspects of semantics, including ontologies and relationships.

The search engine can be further enhanced by ranking the services found in decreasing order of relevance to the user according to his/her request.

References

[1] A4 Methodology. http://www.dcs.warwick.ac.uk/research/hpsg/A4/A4.html

[2] Bellwood, T., 2002. Bellwood, T. et all; UDDI Version 2.04 API Specification, July 2002, http://uddi.org/

[3] Box, D., 2000. Box, D. et all; SOAP 1.1, May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[4] Brooke, J., Fellows, D., Garwood, K. and Goble C., "Semantic matching of Grid Resource Descriptions," In Proceedings of the European Across Grids Conference, 2004, http://www.grid-interoperability.org/semres.pdf

[5] Buyya, R. and Venugopal, S., "The gridbus toolkit for service oriented grid and utility computing: An overview and status report," presented at the 1st IEEE Int.Workshop Grid Economics and Business Models (GECON 2004), Seoul, Korea, Apr. 23, 2004, http://www.gridbus.org/papers/gridbus2004.pdf

[6] Chen, L, Shadbolt, N, Goble, C, Tao, F, Puleston, C, and Cox, S.J., "Semantics-Assisted Problem Solving on the Semantic Grid," Computational Intelligence, Vol.21, No.2, 2005, pp.157-176, http://www.geodise.org/files/Papers/ComputationalIntelligencePaper.pdf

[7] Christensen, E., Curbera, F., Meredith, G. and Weerawarana., S., "Web Services Description Language (WSDL) 1.1," W3C, Note 15, 2001, www.w3.org/TR/wsdl

[8] Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C., "Grid Information Services for Distributed Resource Sharing," In 10th IEEE International Symposium on High Performance Distributed Computing, (2001), IEEE Press, 181-194, http://www.globus.org/alliance/publications/papers/MDS-HPDC.pdf

[9] EU DataGrid Project, March 2004. http://www.eu-datagrid.org

[10] Ferreira, L., Bieberstein, N., Berstis, V., Armstrong, J., "Introduction to Grid Computing with Globus," Redbook, IBM Corp., http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf

[11] Fitzgerald, S., Foster, I., Kesselman, C., Laszewski, G. Von, Smith W., and Tuecke S., "A Directory Service for Configuring High-Performance Distributed

Computations," In Proceedings of the 6th IEEE Symp. on High-Performance Distributed Computing, pages 365-375. IEEE Computer Society, 1997,

[12] Foster, I. and Kesselman, C., "The Grid: Blueprint for a New Computing Infrastructure," San Mateo, CA: Morgan Kaufmann, 1999.

[13] Foster, I., Kesselman C., Nick J.M. and Tuecke S. "Grid services for distributed system integration," Computer, vol. 35, no. 6, pp. 37–46, June 2002, http://www.globus.org/alliance/publications/papers/ieee-cs-2.pdf

[14] Foster, I., Kesselman C., Nick, J.M., and Tuecke, S., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002, http://www.globus.org/alliance/publications/papers/ogsa.pdf

[15] Foster, I., Kesselman, C. and Tuecke, S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001. http://www.globus.org/research/papers/anatomy.pdf

[16] Foster, I. (Ed.), Kishimoto, H. (Ed.), Savva, A. (Ed.), Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., and Von Reich, J., "The Open Grid Services Architecture Version 1.0," Global Grid Forum, Lemont, Illinois, USA, January 29, 2005. GFD-I.030, http://www.gridforum.org/documents/GFD.30.pdf

[17] Foster, I., Roy A., Sander, V., "A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation," 8th International Workshop on Quality of Service, 2000.

[18] Foster, I., Roy A., Sander, V., Winkler, L., "End-to-End Quality of Service for High-End Applications," Technical Report, 1999.

[19] Globus Project, 2004. http://www.globus.org

[20] Globus WS-Resource Framework (WSRF), 2004. http://www.globus.org/wsrf

[21] Goble C. and Roure, D. De. "The semantic web and grid computing in Real World Semantic Web Applications," Frontiers in Artificial Intelligence and Applications, Kashyap, V. and Shklar, L., Eds. Amsterdam, The Netherlands: IOS, 2002, vol. 92.

[22] GridBus Project at Grid Computing and Distributed Systems (GRIDS) Laboratory, 2004 , http://www.gridbus.org/

[23] Grid Interoperability Project GRIP, February 2004. http://www.grid-interoperability.org/

[24] Grid Market Directory http://www.gridbus.org/gmd/

[25] Harchol-Balter, M., Leighton, T. and Lewin, D., "Resource discovery in distributed networks," ACM Symposium on Principles of Distributed Computing, May 1999, pp. 229-237.

[26] Harth, A., Decker, S., He, Y., Tangmunarunkit, H., Kesselman C., "A semantic matchmaker service on the grid," World Wide Web Conference 2004: 326-327.

[27] History.http://gridcafe.web.cern.ch/gridcafe/whatisgrid/dream/powergrid.html

[28] Heker, S., Reynolds, J., and Weider, C.," Technical overview of directory services using the X.500 Protocol," RFC 1309, FY14, 03/12 92.

[29] Huang, Y., Venkatasubramanian, N., "QoS-based resource discovery in Intermittently available environments," Proc. of 11th IEEE International Symposium on High Performance Distributed Computing, pp: 50 -59, HPDC-11, 2002.

[30] Iamnitchi, A. and Foster, I., "On Fully Decentralized Resource Discovery in Grid Environments". Lecture Notes in Computer Science, 2242:51-62, 2001.

[31] Iamnitchi, A., Foster, I., Nurmi, D. C., "A Peer-to-Peer Approach to Resource Discovery in Grid Environments," Proc. of the 11th Symposium on High Performance Distributed Computing, Edinburgh, UK, 2002.

[32] Introduction to grid computing. http://www.lip.pt/computing/projects/EGEE/public_gridintro_en.htm

[33] Joseph, J., Ernest, M., Fellenstein, C., "Evolution of grid computing architecture and grid adoption models," IBM Systems Journal, Vol 43, No. 4, 2004.

[34] Krauter, K., Buyya, R. and Maheswaran, M., "A Taxonomy and Survey of Grid Resource Management Systems", Software Practice and Experiance, Vol. 32, No. 2, Feb. 2002, pp. 135-164.

[35] Laszewski, G. von and Foster, I. "Usage of LDAP in Globus." http://www.globus.org/ pub/globus/papers/ldap_in_globus.pdf

[36] Li, L. and Horrocks, I, "A Software Framework for Matchmaking Based on Semantic Web Technology", In Proceedings of the 12th international conference on World Wide Web (WWW'03), Budapest, Hungary, May 2003.

[37] Lord, P., Alper P., Wroe, C. and Goble, C., "Feta: A lightweight architecture for user oriented semantic service discovery", In Proceedings of The Semantic Web: Research and Applications: Second European Semantic Web Conference (ESWC 2005), Heraklion, Crete.

[38] Maheswaran, M. and Krauter, K. "A Parameter-based approach to resource discovery in Grid computing systems", 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000), December 2000, Bangalore, India.

[39] Monitoring and Discovery System in Globus Toolkit. http://www.globus.org/toolkit/mds/

[40] Nabrzyski, J., Schopf, J., and Weglarz, J., editors. "Grid Resource Management: State of the Art and Future Trends", volume 64 of International Series in Operations Research & Management Science. Kluwer Academic Publishers, Springer, 1st edition, Nov. 2003.

[41] Nitzberg, B. and Schopf, J.M., "Current Activities in the Scheduling and Resource Management Area of the Global Grid Forum", in: D.G. Feitelson, L. Rudolph and U. Schwiegelshohn, Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 2537, Springer Verlag 2002. http://www-unix.mcs.anl.gov/~schopf/Pubs/lcns02.pdf

[42] Plale, B., Dinda, P., and Laszewski, G. Von., "Key Concepts and Services of a Grid Information Service", In Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems (PDCS 2002), 2002.

[43] Raman, R., Livny, M. and Solomon, M., "Matchmaking: An extensible framework for distributed resource management", Cluster Computing: The Journal of Networks, Software Tools and Applications, 2:129-138, 1999.

[44] Raman, R., Livny, M. and Solomon, M., "Matchmaking: Distributed Resource Management for High Throughput Computing", Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago, IL.

[45] Reynolds, J., and Weider, C., “Executive introduction to directory services using the X.500 protocol,” RFC 1308, FYI 13, 03/12 92.

[46] Roure, D. De, Jennings, N., Shadbolt, N., “Research Agenda for the Semantic Grid: A Future e-Science Infrastructure”, UK e-Science Programme Technical Report Number UKeS-2002-02.

[47] Simple Object Access Protocol (SOAP) 1.1. W3C, Note 8, 2000.

[48] Smith, W. and Gunter D., “Simple LDAP Schemas for Grid Monitoring”, Global Grid Forum, GWD-Perf-13-1, June 2001.

[49] Sun N1 grid Engine 6 http://docs.sun.com/app/docs/doc/817-6117?q=N1GE

[50] Tangmunarunkit, H., Decker, S. and Kesselman,.C., “Ontology Based Resource Matching in the Grid - The Grid Meets the Semantic Web”. In Proceedings of the 2nd International Semantic Web Conference, pages 706–721, 2003.

[51] Tierney, B. et al., "A Grid Monitoring Architecture," Global Grid Forum, Lemont, Illinois, U.S.A., January 2002. http://www.gridforum.org/documents/GFD.7.pdf

[52] UDDI: Universal Description, Discovery and Integration. www.uddi.org

[53] W3C. Resource Description Framework (RDF), 2004. http://www.w3.org/RDF

[54] W3C. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, August 3, 2004. W3C Working Draft.

[55] W3C. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, August 3, 2004. W3C Working Draft http://www.w3.org/TR/2004/WD-wsdl20-20040803

[56] Yeong, W., Howes, T. and Kille S. “Lightweight Directory Access Protocol”. IETF, RFC 1777, 1995. http://www.ietf.org/rfc/rfc1777.txt

[57] Younas, M., Chao, K-M., Anane, R., Yan, S.Y, Lovett, P. J., Godwin, A. N., "Grid Services Mediation," aina, pp. 879-884, 19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers), 2005.

[58] Yu, J., Venugopal, S. and Buyya, R., “A Market-Oriented Grid Directory Service for Publication and Discovery of Grid Service Providers and their Services”, Journal of Supercomputing, Kluwer Academic Publishers, USA, 2005, http://www.gridbus.org/papers/gmd.pdf

Appendix A

Installation of N1 Grid Engine 6

1. Install Solaris 10
2. Create the installation directory as sge-root in which contents of the distribution media will be loaded.
3. Install the binaries for all binary architectures that are to be used by any of your master, execution, and submit hosts in your grid engine system cluster.

   The pkgadd Method

   The pkgadd format is provided for the Solaris Operating System. To facilitate remote installation, the pkgadd directories are also provided in zip files.

   Install the following packages:

   - SUNWsgeec – Architecture independent files
   - SUNWsgeed – Documentation
   - SUNWsgee – Solaris (SPARC platform) 32-bit binaries for Solaris 7, Solaris 8, and Solaris 9 Operating Systems
   - SUNWsgeex – Solaris (SPARC platform) 64-bit binaries for Solaris 7, Solaris 8, and Solaris 9 Operating Systems
   - SUNWsgeei — Solaris (x86 platform) binaries for Solaris 8 and Solaris 9 Operating Systems
   - SUNWsgeeax - Solaris (x64 platform) binaries for Solaris 10 Operating System
   - SUNWsgeea - Accounting and Reporting Console (ARCo) packages for the Solaris and Linux Operating systems.

   At the command prompt, type the following commands, responding to the script questions.

   # cd *cdrom_mount_point*/N1_Grid_Engine_6u4

   # pkgadd -d ./Common/Packages/SUNWsgeec

   # pkgadd -d ./Docs/Packages/SUNWsgeed

   # pkgadd -d ./Solaris_sparc/Packages/SUNWsgee

pkgadd -d ./Solaris_sparc/Packages/SUNWsgeex

pkgadd -d ./Solaris_x86/Packages/SUNWsgeei

pkgadd -d ./Solaris_x64/Packages/SUNWsgeeax

4. Install Grid Engine software interactively.

   - Install Master host

     1. Log in to the master host as root.

     2. Ensure that the $SGE_ROOT environment variable is set by typing:

     # echo $SGE_ROOT

        - If the $SGE_ROOT environment variable is not set, set it now, by typing:

          # SGE_ROOT=*sge-root*; export SGE_ROOT

     3. Change to the installation directory.

        - If the directory where the installation files reside is visible from the master host, change directories (cd) to the installation directory *sge-root*, and then proceed to Step 4.
        - If the directory is not visible and cannot be made visible, do the following:

          a. Create a local installation directory, *sge-root*, on the master host.

          b. Copy the installation files to the local installation directory *sge-root* across the network (for example, by using ftp or rcp).

          c. Change directories (cd) to the local *sge-root* directory.

     4. Type the install_qmaster command.

        This command starts the master host installation procedure. We are asked several questions, and may need to run some administrative actions.

        % ./install_qmaster

     5. Choose an administrative account owner.

        sgeadmin.

     6. Verify the *sge-root* directory setting.

     7. Enter the name of your cell.

        CoEPDC

     8. Specify a spool directory.

        /opt/n1ge6/default/spool/qmaster is the qmaster spool directory by default.

9. Specify whether you want to use classic spooling or Berkeley DB.
10. Enter a group ID range 20000-20100
11. Verify the spooling directory for the execution daemon.
    Default: [/opt/n1ge6/default/spool]
12. Enter the email address of the user who should receive problem reports.
    msingh@tiet.ac.in
13. Identify the hosts that you will later install as execution hosts.

- **Install Execution host**

1. Log in to the execution host as root.
2. As you did for the master installation, either copy the installation files to a local installation directory *sge-root* or use a network installation directory.
3. Ensure that you have set the $SGE_ROOT environment variable by typing:
   # echo $SGE_ROOT
   If the $SGE_ROOT environment variable is not set, set it now, by typing:
   # SGE_ROOT=*sge-root*; export SGE_ROOT
4. Change directory (cd) to the installation directory, *sge-root*.
5. Verify that the execution host has been declared on the administration host.
   # qconf -sh
6. Run the install_execd command.
   % ./install_execd
   This command starts the execution host installation procedure.
7. Verify the *sge-root* directory setting.
8. Enter the name of your cell. CoEPDC
9. The install script checks to see if the admin user already exists. If the admin user already exists, the script continues uninterrupted otherwise the script shows a screen where you must supply a password for the admin user. After the admin user is created, press Enter to continue with the installation.
10. Specify whether you want to use a local spool directory.
11. Specify whether you want execd to start automatically at boot time.
    Specify a queue for this host.

Appendix B

Installation of GT4

GT4 is downloadable from http://www.globus.org/toolkit/downloads/4.0.0/

In our lab we installed GT4 on Red Hat Linux 9 so we downloaded gt4.0.0-all-source-installer.tar.gz from the above mentioned site.

1. Before Installation

- Create a user named 'globus' in the local machine. If installing in the head node of the cluster and username is shared between nodes, create a common user to all the nodes.
- Select a partition with at least 500 MB of space. And create a directory and set its access rights so that 'globus' user has the full control over it.
- If installing into a NFS mount, then it is easier to share the same Globus installation among various systems.
- Set the basic environment variable GLOBUS_LOCATION to the directory selected for installing GT4.
- Create a separate folder for Compiled version of GT4 and extract GT4 into it. It can be used in future for some maintenance work.
- Using csh is the best option; bash will also be needed for some scripts.

2. Prerequisites

The following table provides the details of the packages that need to be installed before the installation of GT4.

Direct Downloads *	Website	Compulsory Requirement	Description
GNU tar,sed,Make,gcc	http://www.gnu.org	Yes	Basic tools
Ant 1.5.1+	http://ant.apache.org/	Yes	platform independent build tool
J2SE 1.4.2+	http://java.sun.com/j2se	Yes	java standard edition. this is not directly downloadable
zlib 1.1.4+	http://www.gzip.org	Yes	compression libs required by GPT
sudo	http://www.courtesan.com/sudo	Yes	a tool that provides normal users to execute super user level commands, used by GRAM
JDBC compliant database	http://www.postgresql.org	Yes	recommended is postgresql 7.1+ for RFT, CAS etc
IODBC	http://www.iodbc.org	-	for RLS(Replica Location Service)
Tomcat	http://jakarta.apache.org/tomcat/	-	optional for runtime needs
Junit	http://www.junit.org	-	Needed by ant for unit testing.
Mail server	-	-	Needed for sending and receiving certificates.

* **Direct Downloads:** These are downloads that can be attained directly by clicking the link.

2.1 Apache® Ant

Ant is a platform independent build tool required in both Windows and UNIX based platform before installing GT4. All the compilation of WSComponents is based on Ant. Junit package is needed along with Ant for some tests that may follow. If not installed, that particular test will fail.

1. Download the Binary Distribution of Ant

2. Extract it into the folder where to install it.

3. set environment ANT_HOME to /usr/local1/apache-ant (the installed location)

4. Copy only the junit.jar from Junit distribution into Ant 'lib' Folder

5. Example

```
root# cd /usr/local
root# mkdir apache-ant
root# tar –zxvf apache-ant.tar.gz
```

2.2 Sun J2SE

J2SE is important for all the java based services.

1. Download J2SE Binary for your platform and do the simple step of extract into the folder.

2. Set environment JAVA_HOME to the newly installed J2SE Root.(preferably in .cshrc)

3. Example,

```
root# cd /usr/local
root# mkdir j2sdk
root# cd j2sdk
root# tar –zxvf ~/j2sdk-linux.tar.gz
```

2.3 PostgreSQL

PostgreSQL is an open source implementation of RDBMS system supporting TCP/IP communication. You can just download and install the postgresql base distribution instead of downloading the full collection.

1. During configure, installing without the support of readline will not cause any problems. If configure script gives out error from unavailable readline package, just override it using –without-readline option.

2. do make and make install

3. Example

```
root# cd /usr/local
root# mkdir pgsql
root# cd $HOME
```

root# tar –zxvf postgresql-base-8.0.3.tar.gz

root# cd postgresql-base-8.0.3

root# ./configure –prefix=/usr/local/pgsql

root# make

root# make install

Configuring PostgreSQL needs the following steps:

1. add a new user postgres with normal privileges
2. Create a Data Directory to hold the database,

    # mkdir /usr/local/pgsql/data

    # chown postgres /usr/local/pgsql/data

    # su – postgres
3. set environment PGDATA = /usr/local/pgsql/data and also update the PATH variable to $PATH:/usr/local/pgsql/bin in .cshrc
4. $(post)/sbin/initdb . This command creates a file postgresql.conf in the data directory.
5. If configuring for inetd, Create new file in /etc/init.d named postgresql

    - type the following into that file

    su – postgres –c "/usr/local/pgsql/bin/pg_ctl start –o "-i" –l logfile –D /usr/local1/pgsql/data"

    Note: -i option is needed to accept TCP/IP connections to database.
6. create a link in the directory /etc/rc3.d as follows

    # ln –s ../init.d/postgresql S99postgresql
7. Make postgresql as an executable using these commands,

    # chmod u+x postgresql

    # chown postgres postgresql

2.4 sudo

sudo is used to run commands that require Super User privilege by the ordinary user itself. It is the important requirement for WS-GRAM. So, configuring sudo is very important for successful execution of jobs using WS-GRAM. Sudo can be installed as follows

1. Download the binary. And Extract it into appropriate folder

2. set PATH if needed

3. Type $(sudo)/sbin/visudo and add the following entry. It will open /etc/sudeors file. Don't use vi to edit this file.

```
# Globus GRAM entries
globus ALL=(username1,username2) NOPASSWD:
/opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute –g
/etc/grid-security/grid-mapfile
/opt/globus/GT4.0.0/libexec/globus-job-manager-script.pl *
globus ALL=(username1,username2) NOPASSWD:
/opt/globus/GT4.0.0/libexec/globus-gridmap-and-execute –g
/etc/grid-security/grid-mapfile
/opt/globus/GT4.0.0/libexec/globus-gram-local-proxy-tool *
```

4. visudo itself reports if there are any error in the entries.

3. Configuring and Installing

After the tar file is downloaded do the following steps,

globus$ cd globus_compiled

globus$ tar –zxvf gt4.0.0-all-source-installer.tar.gz

globus$ ./configure –-prefix=$GLOBUS_LOCATION

globus$ make | tee build.log

Check out for errors if any after make.

If required, login as root

root# make install

4. Setting appropriate paths

Before getting into configuring of globus it is required that all paths are set correctly.

Sample .cshrc file:

set path = (/usr/local1/sudo/sbin /usr/local/mpich-1.2.6/globus2/bin /usr/local/bin /usr/local1/sudo/bin /usr/local/pgsql/bin /bin /usr/ccs/bin /usr/local/ssl/bin

```
/usr/sbin /usr/local/sbin \
. $path)
setenv JAVA_HOME /usr/local/J2SDK/j2sdk
setenv
MANPATH :/usr/local1/gt4.0.0/man:/usr/local/man:/usr/share/man:/u
sr/local1/sudo/man:/usr/local/pgsql/man
setenv JAVA_HOME /usr/local/J2SDK/j2sdk
setenv ANT_HOME /usr/local1/apache-ant
setenv CLASSPATH /usr/local1/junit:junit.jar
setenv GLOBUS_LOCATION /usr/local1/globus
setenv GLOBUS_HOSTNAME `hostname`
setenv PGDATA /usr/local1/pgsql/data
set path = ($ANT_HOME/bin $path $JAVA_HOME/bin $ANT_HOME/bin)
setenv GLOBUS_OPTIONS –Djava.security.egd=file:/dev/urandom
source $GLOBUS_LOCATION/etc/globus-user-env.csh
```

The GLOBUS_OPTIONS environment variable is used to set the container startup options that are used when globus-start-container command is used.
ANT_HOME environment variable is used to specify the ant installed location and JAVA_HOME is used to specify the root of the J2SDK installed location.
PGDATA must point to PostgreSQL data directory.
GLOBUS_HOSTNAME variable is optional and it is useful only when you have some hostname problems.

Appendix C

Installation of Condor on Windows 2000

- **Installation Requirements**
  - Condor for Windows requires Windows 2000 (or better) or Windows XP.
  - 50 megabytes of free disk space is recommended. Significantly more disk space could be desired to be able to run jobs with large data files.
  - Condor for Windows will operate on either an NTFS or FAT filesystem. However, for security purposes, NTFS is preferred.
  - Administrator privileges

- **Installation Procedure**

Download Condor, and start the installation process by running the file (or by double clicking on the file). The Condor installation is completed by answering questions and choosing options within the following steps.

STEP 1: License Agreement.

You are asked to agree to the license. Answer yes. After agreeing to the license terms, the next Window is where fill in your name and company information, or use the defaults as given.

STEP 2: Condor Pool Configuration.

The Condor installation will require different information depending on whether the installer will be creating a new pool, or joining an existing one.

If you are creating a new pool, the installation program requires that this machine is the central manager. For the creation of a new Condor pool, you will be asked some basic information about your new pool:

Name of the pool

hostname

of this machine.

Size of pool

If you are joining an existing pool, all the installation program requires is the hostname of the central manager for your pool.

STEP 3: This Machine's Roles.

This step is omitted for the installation of Personal Condor.

Each machine within a Condor pool may either submit jobs or execute submitted jobs, or both submit and execute jobs. This step allows the installation on this machine to choose if the machine will only submit jobs, only execute submitted jobs, or both. The common case is both, so the default is both.

STEP 4: Where will Condor be installed?

The next step is where the destination of the Condor files will be decided. It is recommended that Condor be installed in the location shown as the default in the dialog box: C: \Condor.

STEP 5: Where is the Java Virtual Machine?

While not required, it is possible for Condor to run jobs in the Java universe. In order for Condor to have support for java, you must supply a path to java.exe on your system. To disable the Java universe, simply leave this field blank.

STEP 6: Where should Condor send e-mail if things go wrong?

Specify the e-mail address and the SMTP relay host of this administrator.

STEP 7: The domain.

This step is omitted for the installation of Personal Condor.

Enter the machine's accounting (or UID) domain.

STEP 8: Access permissions.

This step is omitted for the installation of Personal Condor.

Machines within the Condor pool will need various types of access permission. The three categories of permission are read, write, and administrator.

STEP 9: Job Start Policy.

Condor will execute submitted jobs on machines based on a preference given at installation. Three options are given, and the first is most commonly used by Condor pools. This specification may be changed or refined in the machine ClassAd requirements attribute.

The three choices:

After 15 minutes of no console activity and low CPU activity.

Always run Condor jobs.

After 15 minutes of no console activity.

STEP 10: Job Vacate Policy.

This step is omitted if Condor jobs are always run as the option chosen in Step 9.

If Condor is executing a job and the user returns, Condor will immediately suspend the job, and after five minutes Condor will decide what to do with the partially completed job. There are currently two options for the job.

The job is killed 5 minutes after your return.

The job is suspended immediately once there is console activity. If the console activity continues, then the job is vacated (killed) after 5 minutes. Since this version does not include check-pointing, the job will be restarted from the beginning at a later time. The job will be placed back into the queue.

Suspend job, leaving it in memory.

The job is suspended immediately. At a later time, when the console activity has stopped for ten minutes, the execution of Condor job will be resumed (the job will be unsuspended). The drawback to this option is that since the job will remain in memory, it will occupy swap space. In many instances, however, the amount of swap space that the job will occupy is small.

So which one do you choose? Killing a job is less intrusive on the workstation owner than leaving it in memory for a later time. A suspended job left in memory will require swap space, which could possibly be a scarce resource. Leaving a job in memory, however, has the benefit that accumulated run time is not lost for a partially completed job.

STEP 11: Review entered information.

Check that the entered information is correctly entered. You have the option to return to previous dialog boxes to fix entries.

Appendix D
Installation of Alchemi

The steps for installing various Alchemi Components are given below. It includes the installation, configuration and operations for each of the components.

1. Alchemi Manager

Installation

The Alchemi Manager can be installed in two modes
- As a normal Windows desktop application
- As a windows service. (supported only on Windows NT/2000/XP/2003)

- To install the manager as a windows application, use the Manager Setup installer. For service mode installation use the Manager Service Setup. The configuration steps are the same for both modes. In case of the service-mode, the "Alchemi Manager Service" installed and configured to run automatically on Windows start-up. After installation, the standard Windows service control manager can be used to control the service. Alternatively the Alchemi ManagerServiceController program can be used. The Manager Service controller is a graphical interface, which is exactly similar to the normal Manager application.
- Install the Manager via the Manager installer. Use the sa password noted previously to install the database during the installation.

Configuration & Operation

- The Manager can be run from the desktop or Start -> Programs -> Alchemi -> Manager ->
- Alchemi Manager. The database configuration settings used during installation automatically appear when the Manager is first started.
- Click the "Start" button to start the Manager.
- When closed, the Manager is minimised to the system tray.

2. Cross Platform Manager

Installation

- Install the XPManager web service via the Cross Platform Manager installer.

Configuration

- If the XPManager is installed on a different machine that the Manager, or if the default port of the Manager is changed, the web service's configuration must be modified. The XPManager is configured via the ASP.NET Web.config file located in the installation directory (wwwroot\Alchemi\CrossPlatformManager by default):

```
<appSettings>
<add key="ManagerUri" value="tcp://localhost:9000/Alchemi_Node" />
</appSettings>
```

Operation

- The XPManager web service URL is of the format

  http://[host_name]/[installation_path]
- The default is therefore

  http://[host_name]/Alchemi/CrossPlatformManager
- The web service interfaces with the Manager. The Manager must therefore be running and started for the web service to work.

3. Executor

Installation

The Alchemi Executor can be installed in two modes

- As a normal Windows desktop application

- As a windows service. (Supported only on Windows NT/2000/XP/2003)

- To install the executor as a windows application, use the Executor Setup installer. For service mode installation uses the Executor Service Setup. The configuration

steps are the same for both modes. In case of the service-mode, the “Alchemi Executor Service” installed and configured to run automatically on Windows start-up. After installation, the standard Windows service control manager can be used to control the service. Alternatively the Alchemi ExecutorServiceController program can be used. The Executor service controller is a graphical interface, which looks very similar to the normal Executor application.

- Install the Executor via the Executor installer and follow the on-screen instructions.

Configuration & Operation

The Executor can be run from the desktop or Start -> Programs -> Alchemi -> Executor -> Alchemi Executor.

The Executor is configured from the application itself.

You need to configure 2 aspects of the Executor:

- The host and port of the Manager to connect to.
- Dedicated / non-dedicated execution. A non-dedicated Executor executes grid threads on a voluntary basis (it requests threads to execute from the Manager), while a dedicated Executor is always executing grid threads (it is directly provided grid threads to execute by the Manager). A non-dedicated Executor works behind firewalls.

Click the "Connect" button to connect the Executor to the Manager.

Papers Communicated / Accepted / Published

1. Shruti Goel, Ms. Inderveer Chana, "**Resource Discovery in a Grid Environment**", AICTE sponsored National Seminar on "Information & Communication Technology Recent Advances & Applications" **(ICT-2006)**, Seth Jai Prakash Mukand Lal Institute of Engg. & Tech., Radaur, India, February 9-11, 2006 **(Published in Proceedings).**