Grid Resource Discovery using Web Services

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science & Engineering

By:
Ruchi Garg
(8053216)

Under the supervision of
Dr. (Mrs.) Seema Bawa
Professor & Head, CSED
Thapar University, Patiala

MAY 2007

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Grid Resource Discovery using Web Services"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. (Mrs.) Seema Bawa*

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

(Ruchi Garg)

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

(Dr. (Mrs.) Seema Bawa)
(Supervisor)
Professor & Head
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by :

Dr. (Mrs.) Seema Bawa
Professor & Head
Computer Science & Engineering Department
Thapar University
Patiala

Dr. R.K. Sharma
Dean, Academic Affairs
Thapar University
Patiala

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor **Dr. (Mrs.) Seema Bawa**, Professor & Head, Computer Science & Engineering Department for her immense help, guidance, stimulating suggestions and encouragement all the time. She always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under her supervision.

I am equally grateful to **Dr. Maninder Singh**, Assistant Professor, Computer Science & Engineering Department, a nice person, an excellent teacher and a well-credited researcher, who always encouraged me to keep going with work and always advised me with his invaluable suggestions.

I am most indebted to the P.G coordinator **Mrs. Shivani Goyal** for her help, assistance and suggestions during my work.

I would also like to express my appreciation to the TU Grid Group. The group held weekly meetings on Thursdays to address various issues and share experiences. Thus, it invoked better understanding of the work and acted as a rostrum for information sharing and problem solving.

Finally, my special thanks go to my family for their never-ending love and support. Nothing would have ever been possible without my parent's unconditional love and encouragement.

The most valuable thing I acquired from the two years study in TU is to protect, survive and endure myself in this world with increased confidence and additional faith in my abilities to achieve.

Last but not the least, I would like to thank the lord for his blessings and for driving me with faith, hope and courage in the thinnest of the times.

Ruchi Garg

(8053216)

ABSTRACT

Grids hold the promise to provide global interoperability and interconnectivity involving networks of heterogeneous resources, working in collaboration, to solve problems that cannot be solved using the resources of any single organization.

Opportunistic sharing of Internet-connected resources is a low cost method for obtaining access to unprecedented-scale collections of resources. An essential service in any resource-sharing environment is resource discovery: given a description of the resources desired, a resource discovery mechanism returns locations of resources that match the description.

Web Services are an XML based set of standards and technologies to integrate software application systems using Internet technology. In practice, there is not much difference between the existing Grid and web infrastructure. The implementation of web based Grid is one of the potential solution to Grid infrastructure based problems. This can be done by sharing Grid services across the Grid infrastructure using the underlying web service as vehicles or transporters of these services.

Using web services based resource discovery is a standards based technique for querying resource discovery information which to date, has received surprisingly little analysis as a discovery mechanism for web service based grids.

In this thesis, we proposed, designed and implemented Grid resource discovery using web services on a .NET-based Grid computing framework so as to utilize the computing resources of the vast majority of desktop computers i.e. those running variants of the Microsoft Windows operating system. Using this framework, a language independent and platform independent support is provided via a web services interface. Furthermore we can conclude that on the design principles proposed in this thesis work, a complete Grid Middleware can be implemented on the same lines of which only the Resource Discovery portion has been presented here.

TABLE OF CONTENTS

List of Figures

Chapter 1
Introduction

Distributed computing systems have been in widespread use for many years in research, engineering, industrial and commercial environments. The challenge of Grid computing is to take computing resources that are scattered all over the place to integrate, manage and take advantage of them as if they were one huge, virtual computer. Sharing resources over the internet raises many technical problems, amplified by the potential scale of the resource pool, the heterogeneity of platforms, the diversity in user behaviour and the internet lack of reliability. In these systems, one important functionality is resource discovery, given a resource description, the resource discovery component return the location of one or more resource that fit the description.

1.1 Significance of Resource Discovery

Grid environment is likely to scale to millions of resources shared by hundreds of thousands of participants (institutions and individuals) .No central, global authority will have the means, the incentive, and the participants' trust to administer such a large collection of distributed resources.

- Participation timings of various nodes in Grid will be highly variable. There will be perhaps a large number of stable nodes than in today's P2P systems, but many resources will join and leave the network frequently.
- Resources are of highly diverse types (computers, data, services, instruments, storage space) and characteristics (e.g. operating system). Some resources will be shared following well-defined public policies, such as "available to all from 6pm to 6am". Other resources will participate rather chaotically.
- Technical support is variable. Some participants will benefit from technical support, whereas others will rely on basic tools provided by the community.

For these reasons, Resource Discovery in large-scale Grids can be very challenging.

1.2 Problem Definition

1.2.1 Background

In the traditional distributed computing environment, moving data between different applications and systems often involves defining a rigid format that is agreed under service level agreement, requiring to fit the new software design to the old development framework, hardly building inter-organizational collaboration, and resulting in duplication of work at different places. This tight coupling requires much agreement and shared context among different applications from different organizations in order to be communicable and reliable. Furthermore, the use of distributed computing systems requires high availability, adaptability, and maintainability, and, in order to remain useful, they have to cope with advances in technology, modifications of their operating environment and ever-changing human needs [1].

1.2.2 Current State of the Art

- Current distributed computing technologies do not address the concerns and requirements just listed above.
- Current Internet technologies address communication and information exchange among computers but do not provide integrated approaches.
- Enterprise distributed computer technologies such as CORBA [60] and DCOM [60] enable resource sharing, but in most of the case implementing only within a single organization [2].
- The Open Group's Distributed Computing Environment [60] (DCE) supports secure resource sharing across sites, but it is not flexible.
- Emerging "Distributed Computing" companies seek to harness idle computers on an international scale, but support only highly centralized access to those resources [3, 4, 5, 6].

This comparison will be further discussed in Chapter 4. In summary, current technology either does not accommodate the range of resource types or does not provide the flexibility and control on sharing relationships needed to establish interoperability. [7]

1.3 Methodology Used

It is here that Web Services technologies enter the scenario. In this thesis work, we

have proposed and developed Web Services based Grid Resource Discovery System giving solutions to above mentioned problems. Web Services compliment the next generation of distributed systems. The reasons are briefly discussed below:

•**Loosely coupled, self-describing interface abstraction** Derived from Service Oriented Architecture [52], for distributed heterogeneous system based on technology that is widely available on almost any conventional computer platform. In this respect Web Services are expected to succeed where tightly coupled and other distributed computing technologies have stopped. Web Services are in this sense a logical evolution of, and more pervasive than Component Based Software Architectures, where they do not require a specific component implementation technology, but basically just a standard self-describing (XML) interface on top of "any" implementation.

•**Interoperability** Any Web Service can interact with any other Web Service. SOAP, the standard protocol is supported by all of the major vendors. As Web Services can be written in any language (even COBOL according to Microsoft [53]) and any platform, developers do not need to change their development environments in order to produce or consume Web Services.

•**Ubiquity** Web Services could communicate using HTTP and XML, which are widely acceptable. Therefore, any device, which supports these technologies, can both host and access Web Services.

•**Low barrier to Entry** The concept behind Web Services is easy to understand. And there are a lot of free toolkits from vendors like IBM, Microsoft and Apache open source allow developers to quickly create and deploy Web Services. In addition, some of these toolkits allow pre-existing COM components and Java Beans to be easily exposed as Web Services.

•**Industry Support** All of the major vendors are supporting SOAP and the surrounding Web Services technology. For example, the Microsoft .NET platform supports Web Services, thereby making it very easy for components written in Visual Basic to be deployed as Web Services, and consumed by Web Services written using IBM VisualAge, and vice-versa.

•**Comparable advantage** Web Services use HTTP to be firewall friendly and employs XML as an encoding schema. It offers a free versus fee value proposition with regard to HTTP/SOAP server environments versus ORB frameworks, uses the pervasive Internet concept of URLs to address object identification. Web Services

vendors are actually working actively to prove that SOAP implementations do interoperate. And SOAP format is self-describing contrast with CORBA and DCOM protocols, which are binary and tough to trace.

1.4 Organization of the 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, and various types of Grid systems, Grid architecture, Grid topology, and benefits of Grid.

Chapter 3 describes the Resource Discovery process and various approaches used by this process alongwih it also Resource Discovery mechanisms used in state of art Grid systems.

Chapter 4 is the result of a literature study in the area of Web Services, and introduces some important terminology, definitions and concepts used to discuss about Web Services. Furthermore, it presents some of the current approaches to Web Services and compares these approaches. It also provides some background and discusses related work to Web Services, which discusses recent efforts in developing technologies that enable interoperability.

Chapter 5 captures the problem statement, which deals with developing a web services based resource discovery using .NET framework.

Chapter 6 is all about details of practical demonstration of the Web based Resource discovery in the Grid environment.

Finally future scope of work is being discussed.

Chapter 2

Grid Computing: An Introduction

Grid is indicated as a proposed distributed computing infrastructure for advanced science and engineering. It is distinguished from conventional distributed computing by its focus on large-scale, resource sharing, innovative application and high-performance orientation, which has been explained and defined as *Virtual Organization (VO)[2].* Grid computing defines the sharing clearly as what is shared, who is allowed to share, and the conditions under which sharing occur. The real and specific problem that underlies the Grid concept is coordinated resource[7] sharing and problem solving in dynamic, multi-institutional virtual organization. The establishment, management, and exploitation of dynamic, cross-organizational VO sharing relationships require Grid architecture, in which protocols, services, application programming interfaces, and software development kits are categorized according to their roles in enabling resource sharing.

The Grid will have advanced applications running on a tremendously powerful supercomputing facility to aid the radiologist, or any other physician, in analyzing precisely what's going on in a particular patient. If multiple physicians, perhaps scattered across the country (or even spread all over the world) need to consult and view a particular mammogram, this Grid will give them access to the information as if they were in the same room of the same hospital. That's the example of the kinds of virtual collaboration and virtual organization that Grid computing is trying to address. For instance, brain mapping research, which is absolutely critical to better understand and treat Alzheimer's, schizophrenia, Parkinson's, multiple sclerosis and a whole set of neurological diseases that people don't fully understand now. The problem with brain mapping research is that each MRI of the brain can take up a huge amount of storage. And physicians need lots of images of different brains for comparison; and need to do this as a collaborative effort between multiple centers, each of which can share the data and analyzing tools with others.

Desired attributes for distributed computing on the Grid:

•Allow computing resources to be accessed and shared across a heterogeneous wide area network, which provides language independent; platform independent; support

resource discovery, access, brokering; access not only software, but also underlying computing resources.

•Deliver qualities of service transparently, which includes seamless integration with existing IT resources; distributed application workload management; scalable performance; distributed security model across administrative domains; logging; problem determination; health monitoring, checkpoint, coordinated fail over.

Grid computing, emerging from the science and technical community, generates new e-business infrastructure opportunities with expansion of resource sharing and problem solving on the Internet. This allows all researchers to build and participate in virtual organizations. Grid computing also highlights the need for open standards and for self-managing technologies.

2.1 Distributed Computing

Distributed computing [9-13] is often used to describe a type of computing in which different computing nodes can simultaneously run an application located on different computers that are connected by a communication network. The main motivation for constructing distributed systems is to obtain large-scale resource sharing at affordable cost. Advances in networking technology and computational infrastructure make it possible to construct large-scale high performance distributed computing environments that provide dependable, consistent and pervasive access to high-end computational and heterogeneous resources despite geographical distribution of both resources and users. Grid computing has been widely seen as a step beyond the conventional distributed computing, incorporating pervasive high-bandwidth, high-speed computing, intelligent sensors and large scale database into a seamless pool of managed and brokered resources. These kinds of large-scale high performance distributed services have recently received substantial interest from both research as well as industrial point of view.

Hence, Grid computing is a form of distributed computing that involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations. Grid technologies promise to change the way organizations tackle complex computational problems.

2.1.1 How does Grid Computing surpass Distributed Computing?

Distributed applications already exist, but they tend to be specialized systems intended for a single purpose or user group. Grids go further and take into account:

- Different kinds of resources: Not always the same hardware, data and applications
- Different kinds of interactions: User groups or applications want to interact with Grids in different ways.
- Dynamic nature: Resources and users added/removed/changed frequently.

2.2 History of Grid

Many of the basic ideas behind the Grid have been around in one form or other throughout the history of computing. For example, one of the novel ideas of the Grid is sharing computing power. Nowadays, where most people have more than enough computing power on their own PC, sharing is unnecessary for most purposes. But 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.

In 1965 the developers of an operating system called Multics[14], an ancestor of Unix, which in turn is an ancestor of Linux - a popular operating system today presented a vision of computing as a utility - in many ways uncannily like the Grid vision today. Access to the computing resources was envisioned to be exactly like water, gas and electricity - something that the client connects to and pays for according to the amount of use.

So there is a certain amount of "reinventing the wheel" going on in developing the Grid. However, each time the wheel is reinvented, it is reinvented in a much more powerful form, because computer processors, memories and networks improve at exponential rates which are associated with Moore's law[14].

Because of the huge improvements of the underlying hardware typically more than a factor of 100x every decade, it is fair to say that reinvented wheels are qualitatively different solutions, not just small improvements on their predecessor.

2.3 Grid Types

Grids can be classified on the basis of two parameters

- Grid functionality
- Grid topology

2.3.1 Functional classification

Grid systems can be classified into two categories. Real Grids may be a combination of two of these types. The two categories of Grid systems are described below.

- Computational Grid
- Data Grid

Computational Grid

Computational Grids can be recognized by these primary characteristics:

- Made up of clusters of clusters.
- Enables CPU scavenging to better utilize resources.
- Provides the computational power to process large-scale jobs to satisfy the business requirement for instant access to resources on demand

Data Grid

Data Grids focus on providing secure access to distributed, heterogeneous pools of data. Data Grids also harness data, storage, and network resources located in distinct administrative domains, respect local and global policies governing how data can be used.

2.3.2 Topology Classification

Grids can be built in all sizes, ranging from just a few machines in a department to groups of machines organized as hierarchy spanning the world. Grids can be classified into three categories according to the topology of Grid.

- IntraGrid
- ExtraGrid
- InterGrid

Intra Grid

- There is a single organization.
- Partner integration is not there.
- Only one single cluster.

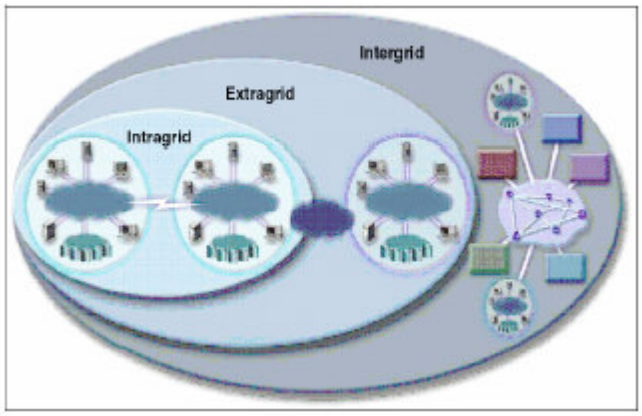


Figure 2.1: Grid Types [59]

Extra Grid

- There are multiple organizations.
- Partner integration is there.
- Multiple clusters

Inter Grid

- There are many organizations.
- There are multiple partners
- Many multiple clusters

2.4 Grid Architecture

The architecture of the Grid is often described in terms of "layers", each providing a specific function. In general, the higher layers are focused on the user (user-centric, in the jargon), whereas the lower layers are more focused on computers and networks (hardware-centric) [8].

Fabric Layer

For all the physical infrastructure of the Grid, including computers and the communication network. It is made up of the actual resources that are part of the Grid, such as computers, storage systems, electronic data catalogues, and even sensors such as telescopes or other instruments, which can be connected directly to the network.

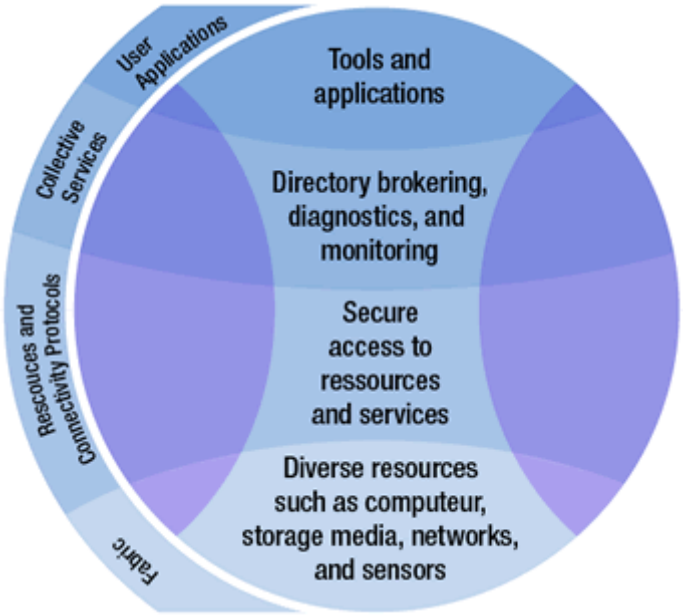


Figure 2.2: Layered Grid Architecture [59]

Resource and connectivity protocols

Resource and connectivity protocols handle all "Grid specific" network transactions between different computers and other resources on the Grid. Grids have to be able to recognize messages that are relevant to them, and filter out the rest. This is done with communication protocols, which let the resources speak to each other, enabling exchange of data, and authentication protocols, which provide secure mechanisms for verifying the identity of both users and resources [8].

Collective services

The collective services are also based on protocols: information protocols, which obtain information about the structure and state of the resources on the Grid, and management protocols, which negotiate access to resources in a uniform way. The services include:

- Keeping directories of available resources updated at all times,
- Brokering resources
- Monitoring and diagnosing problems on the Grid
- Replicating key data so that multiple copies are available at different locations
- Providing membership/policy services for keeping track on the Grid of who is allowed to do what, when [8].

Applications Layer

The highest layer of the structure is the application layer, which includes all different user applications (science, engineering, and business, financial), portals and development toolkits supporting the applications. This is the layer that users of the Grid will "see" [8].

2.5 Benefits of Grid Computing

Some of the benefits of Grid Computing are listed below [1]:

- Provides users around the world with dynamic and adaptive access to unparalleled levels of computing.
- Improved methods for collaborative work.
- Unprecedented Price-to-Performance ratio.
- With the infrastructure provided by the Grid, scientists are able to perform complex tasks, integrate their work and collaborate remotely.
- Grids can lead to savings in processing time.
- Efficient, effective, and economic utilization of available resources.
- Increased availability and reliability of resources.
- Shared access (by multiple users) to large amounts of data.
- Seamless and secure access to large number of geographically distributed resources.
- Reduction in average job response time may occur but an overhead of limited network bandwidth and latency exists [1].

Chapter 3
Resource Discovery in Grid

Generally, *Grid resource management* is defined as the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring Grid resources over time in order to run Grid applications as efficiently as possible[39].

Grid resource management is the process of managing available resources and system workloads accordingly. It is the manner in which resources are allocated, assigned, authenticated, authorized, assured, accounted and authenticated. Resource here includes traditional resources like computing cycles, memory, network bandwidth, space or a storage systems etc that is required by a job for its computation. Resources have their owners who may charge others for using resources. The overall aim of the resource management is to efficiently discover and schedule applications that need to utilize the available resources in heterogeneous and dynamic environments. Therefore, Grid Resource Management can be defined as a process that includes following:

- Identify the requirements
- Matching resources to the applications i.e. Resource discovery
- Reservation, Brokering & Discovery
- Scheduling and Monitoring Grid resources
- Allocating resources
- Management of virtual Organization
- Problem determination & fault management

These resource management scenarios often include resource discovery, resource monitoring, resource inventories, resource provisioning, fault isolation, variety of autonomic capabilities and service level management activities. The promise of Grid computing for effectively harnessing computing resources across organizations is enormous. Resource Broker, a core part of the Grid system, heads the whole hierarchy of maintaining all the service information of the system. Broker performs various processes like discovering, scheduling, co-allocating and evaluating. Out of these

resource discovery is the first and the foremost important process. Broker has to discover the resources before implementing other GRM scenarios.

Resource Discovery

Resource Management system performs resource discovery to obtain information about the available resources. Resources Discovery is systematic process of determining which Grid resource is the best candidate to complete a job with following trade-offs[40].

- In shortest amount of time.
- With most efficient use of resources.
- At minimum cost

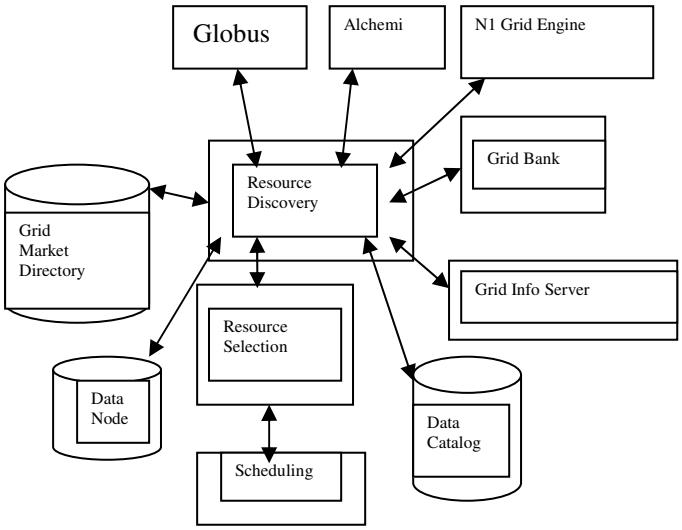


Figure 3.1: Grid Resource Discovery

Resource Discovery schema maintains and queries a resource database known as "status" database. This database is maintained network wide to fulfill the client request information service. Discovery is initiated by a network application to find suitable resources within the Grid. For this it has to interact with the resources individually through agents or it maintains the information about the resources in databases, which are queried w.r.t the application requested.

3.1 Grid Resource Brokering and Discovery

Grid computing infrastructures offer a wide range of distributed resources to applications. To support application execution in the context of the Grid, a Grid Resources Broker is desirable. Grid Resource Brokering is defined as the process of

making scheduling decisions involving resources over multiple administrative domains. This can include searching multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites.

A Grid broker must make resource selection decisions in an environment where it has no control over the local resources. The resources are distributed, and information about the resources is often limited or outdated. A resource broker is a central component in a Grid environment. The purpose of the broker is to dynamically identify, characterize, evaluate, select, allocate and coordinate resources with different characteristics, while at the same time handling constraints of varying nature.

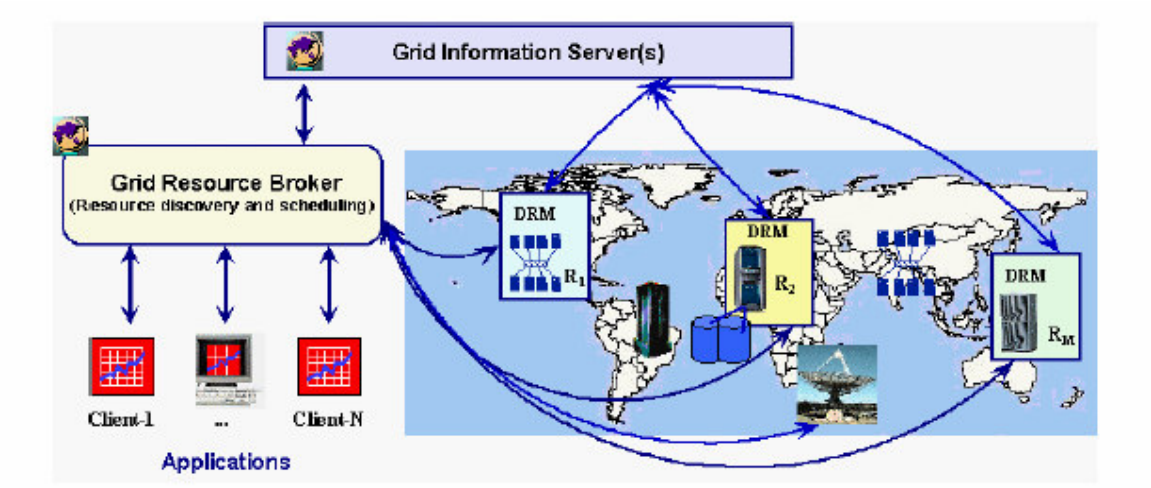


Figure 3.2: Grid Computing at a glance [58]

The main tasks done by Resource Broker are:

- Discover the Resources
- Resource Selection
- Evaluate and Schedule the resources

Resource discovery plays an important role in resource broker. The goal of **Resource Discovery** is to identify a list of authenticated resources that are available for job submission. In order to cope with the dynamic nature of the Grid, a broker needs to have some way of incorporating dynamic state information about the available resources into its decision-making process. This process is termed as resource discovery.

3.2 Resource Discovery Approaches

There are various approaches for resource discovery in Grid environments. Some of these approaches are listed as follows:

3.2.1 Query Based and Agent Based

Resource discovery is either query based or agent based. In a **query based** approach the resource information is queried for resource availability. Most of the contemporary Grid systems follow this approach. Query based system are further characterized depending on whether the query is executed against a distributed database or a centralized database.

On the other hand, **Agent based** approaches send active code fragments across machines in the Grid that are interpreted locally on each machine [41]. Agents can also passively monitor and can distribute resource information either periodically or in response to another agent. Thus agents can mimic a query based resource discovery scheme. The major difference between a query based approach and an agent based approach is that agent based systems allow the agent to control the query process and make resource discovery decisions based on its own internal logic rather than residing upon a fixed function query engine[42,43]. Most agent systems are based on an underlying mobile code environment like java. Agent based resource discovery is inherently distributed.

3.2.2 Peer-to-Peer Approach

Iamnitchi discussed peer-to-peer resource[44,45] discovery in detail. He proposed peer-to-peer resource discovery architecture for a large collection of resources. Different resource discovery problems are there in large distributed resource-sharing environment especially in Grid environment. Four different architectural components and four environments parameter factors are identified, which dominate the performance and design strategies for a resource discovery solution. Using four axes framework, it is possible to design any resource discovery architecture in a Grid. A general purpose query support enabled “unified Peer-to-Peer Database Framework (UPDF)” for large distributed systems has been proposed.

3.2.3 Ontology Description-Based Approach

Ontology refers to a description of a service (resource). A semantic service discovery framework in a Grid environment is proposed. A service matchmaking mechanism [49] based on ontology knowledge was proposed. It is claimed that this matchmaking framework can provide a better service discovery and also can provide close matches. The main idea behind this approach is to advertise the resources. In this approach, service provider registers its service description into the service registry database. When a Grid application sends a request to service directory, matchmaker returns the matches to the service requestor. Requestor chooses the best resource based on the specific need.

3.2.4 Parameter-Based Approach

Different approaches for resource discovery in a Grid system are described .A new concept "Grid potential "[46] is proposed which encapsulates the processing capabilities of different resources in a large network. "Data Dissemination Algorithm" has been proposed. This algorithm follows swamping approach for message distribution. When a message comes to a node, that message gets validated. The validation process depends on three types of dissemination; universal awareness, neighborhood awareness, and distinctive awareness. The performance of "universal awareness", "neighborhood awareness", and "distinctive awareness" dissemination schemes was measured.

3.2.5 Quality of Service (QoS)-based Approach

An algorithm to discover the occasionally available resources in a multimedia environment [47] is proposed. Different policies for a QoS based resource discovery service for a given graph theoretic approaches [48] are proposed. A generalized version of Discovering Intermittently Available Resources (DIAR) algorithm based on occasionally available resources is introduced. The performance of QoS policies based on different time-map strategies in a centralized system is evaluated. Through the experiment it is found that randomized placement strategies and increased server storage can facilitate better performance to discover a particular resource.

3.3 Resource Discovery Algorithms

A number of algorithms have been developed for resource discovery in Grids, including Flooding Algorithm, Swamping Algorithm, Random Pointer Jump Algorithm, Name Dropper Algorithm, Data Flooding Based Data Dissemination Algorithm and Discovering Intermittently Available Resources (DIAR) Algorithm. Some of these algorithms are listed as follows:

3.3.1 The Flooding Algorithm

This algorithm [50] is based on agent-based approach and is widely used by internet routers. Every node acts as a transmitter and receiver and every node tries to send every message to every node of its neighbor. Newly added new edge is not used for any communication. Direct communication exists only in between initially existing set of neighboring edges of the network. The required number of rounds of this algorithm is equivalent to the diameter of the graph. This algorithm can be very slow if not started with a graph, which has small diameter.

3.3.2 The Swamping Algorithm

Agent based approach acts as the base approach for implementing this algorithm. Swamping algorithm [50] is similar to flooding algorithm except this algorithm allows a node to connect not only with the set of initial neighbors but also with all of its current neighbors. The main advantage of this algorithm is this algorithm needs O(log n) rounds to converge to a complete graph and which is irrespective to the initial configuration. The disadvantage is communication complexity of this algorithm grows very quickly.

3.3.3 The Random Pointer Jump Algorithm

In this algorithm [50], in each round, each node contacts with a random neighbor, and then this random neighbor sends all of its neighbors to the sender node. Finally sender neighbor and random neighbor's neighbors get merged. This algorithm claimed that a strongly connected graph with n nodes needs $\Omega(n)$ complexity time to converge to a complete graph.

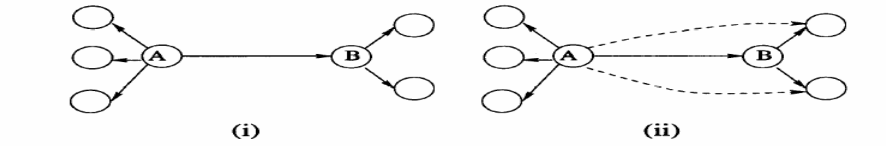


Figure 3.3: The Random Pointer Jump Algorithm [50]

(i) Before the Random Pointer Jump. Node A chooses at random one of its neighbors and opens a connection with it. Here B is the chosen neighbor (ii) After the Random Pointer Jump Node B has passed to node A all of its neighbors, and now A also points to them. The dashed lines indicate newly formed edges.

3.3.4 The Random Pointer Jump with Back Edge Algorithm

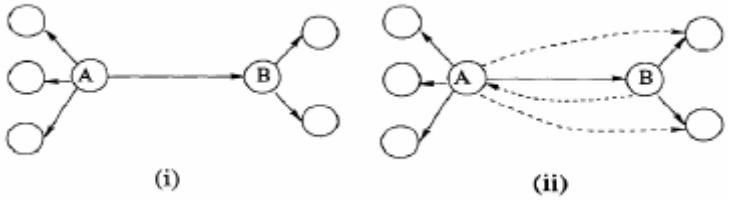


Figure 3.4: The Random Pointer Jump with Back Edge Algorithm [50]

(i) Before the Random Pointer Jump with Back Edge [50]. Node A chooses at random one of its neighbors and opens a connection with it. Here B is the chosen neighbor (ii) After the Random Pointer Jump Node B has passed to node A all of its neighbors, and now A also points to them. Node B is also given a pointer to node A. The dashed lines indicate newly formed edges (Harchol-Balter, 1999) claimed that this algorithm is almost identical to Random Pointer Jump algorithm except every time adding a back edge after performing the pointer jump.

3.3.5 Discovering Intermittently Available Resources (DIAR) Algorithm

This algorithm [50] is used to discover the occasionally available resources in multimedia environment. Different policies for a QoS based resource discovery service for a given graph theoretic approach is defined. In this paper a generalized version of Discovering Intermittently Available Resources (DIAR) algorithm based

on occasionally available resources is introduced. The performance evaluation of QoS policies based on different time-map strategies in a centralized system proves that randomized placement strategies and increased server storage can facilitate better performance to discover a particular resource.

In this section we have reviewed the approaches and existing algorithms for resource discovery in large scale heterogeneous environment. Based on the review work the next section focuses upon the existing gaps that used to be addressed.

Problems with traditional Grid Resource Discovery Approaches

- Traditional Grid resource discovery approaches, with their heavy science focus, typically rely on hosting environments provided by native operating system processes.
- Grid Semantics also usually come from the operating system processes.

Chapter 4
Web Services

Web Services are an XML based set of standards to integrate software application systems using Internet technology. Key element is the capability to publish, find and bind Web Services through public or private registries. Both business services and the actual electronic integration of software services are described and made operationally available in a highly standardized method. Web Services are stated as the latest development in distributed computing on the Internet and are key in the product strategies of most if not all major software vendors today. Web Services are a new, pervasive kind of middleware designed to integrate software systems of a very wide variety. [15]

Web Services are independent of specific programming languages or operating systems. Most important, Web Services rely on pre-existing transport technologies (such as HTTP) and standard data formatting techniques (such as XML) for their implementation. Thus this new emerging technology inherits both standards' advantages and presents a new coupling model that supports loosely coupled collaborations.

One of the main ideas behind Web Services is that applications of the future will be assembled from a collection of network-enabled services. As long as equivalent services are able to advertise themselves to the network in a standard and neutral way, an application could theoretically choose among alternative competing services based on its criteria.

4.1 Web Services Model

4.1.1 Communication Evolution

The way of how participant computers communicate with each other takes place the following four-stage evolution as illustrated in figure 4.1.

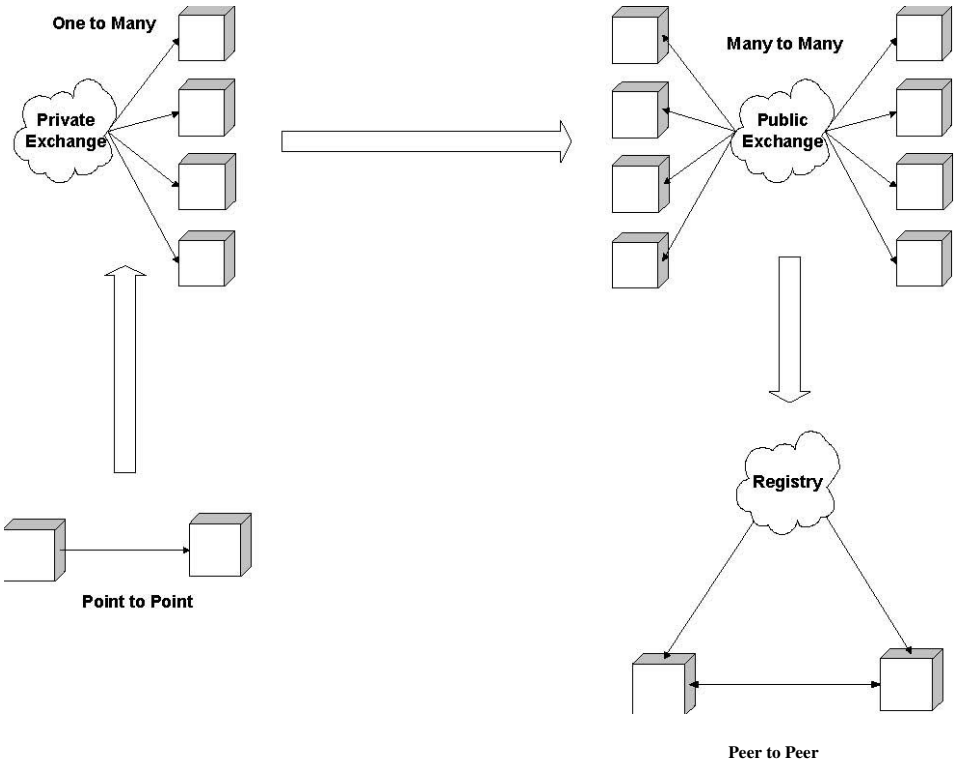


Figure 4.1: Communication Evolution [15]

•Point-to-point communication: namely, one-to-one communication, each single computer talks to another predefined computer;

•One-to-many: via a private exchange server, each computer can talk to many other computers; this is usually taken place in one organization;

•Many-to-many: via a public exchange server, a group of computers can talk to other groups of computers. In the business domain, this is so-called Marketplace model via a broker.

•Peer-to-peer: such as Web Services model where communication is conducted by advertising themselves to the network, so that individual scattered computing nodes communicate with each under the help of private/public registry.

4.1.2 Web Services Component Roles and Operations

There are three components in the Web Services architecture: service provider, service broker and service requester. These parties perform three fundamental operations: publish, find and bind (Figure 4.2) [16]. These concepts are explained as following: (The standards being used in the operations will be explained in details in next section.)

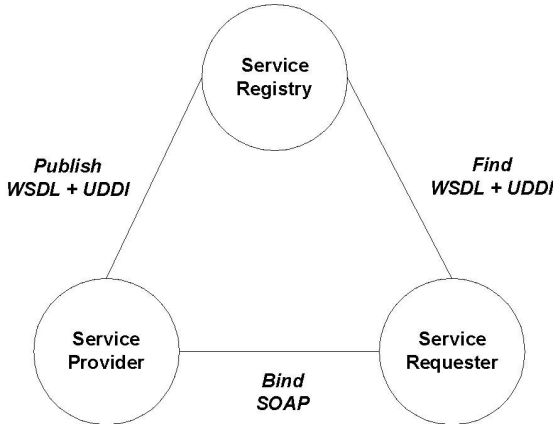


Figure 4.2: Web Services Model [16]

Roles in Web Services architecture:

•**Service provider** From a business perspective, this is the owner of the service. From an architecture perspective, this is the platform that hosts access to the service.

•**Service requester** From a business perspective, this is the business entity that requires certain functions to be satisfied. From an architecture perspective, this is the application that is invoking or initiating an interaction with a service. A program or other Web Services can play this role.

•**Service registry** This is a service description repository where service providers publish their service descriptions and service requester find services and obtain binding information.

Operations in Web Services architecture:

•**Publish** In order to be accessible, a service description needs to be published so that the service requester can find it. The published information can vary from business entity, business description and service description.

•**Find** The service requester retrieves a service description from the service registry in order to find the required business type. The find operation can be involved in two different phases for service requester: at design time to retrieve the service's interface description for program development, and at runtime to retrieve the service's binding and location description for invocation.

•**Bind** The service needs to be invoked. In the bind operation the service requester invokes or initiates an interaction with the service at runtime using the binding details in the service description.

In service-oriented architectures, service descriptions and metadata play a central role in maintaining a loose coupling between service requesters and service providers. The service description, published by the service provider, allows service requesters to bind to the service provider. The service requester obtains service descriptions through a variety of techniques, from the simple "e-mail me the service description" approach to techniques such as Microsoft's DISCO [16] and sophisticated service registries like UDDI.

4.2 Web Services Standards

4.2.1 HTTP and XML - extensible Markup Language

Research and industry has widely accepted HTTP, which is being used everywhere, on almost all platforms. This ubiquity makes HTTP a good choice for an interoperable transport mechanism.

XML is becoming as ubiquitous as HTTP. It stands for extensible Markup Language, which provides a common language for exchange information. It is designed to provide more flexible and adaptable information identification. XML is actually a "meta-language" -- a language for describing languages -- that lets users design their own customized markup languages for different types of documents. Because XML is just text, any application can understand it as long as the application understands the character encoding in use. This makes XML a good choice for describing method invocations in a platform and language-neutral fashion.

Combining HTTP and XML into a single solution gives a whole new level of interoperability. For example, lathered with SOAP that is discussed in next section, clients written in Microsoft Visual Basic can easily invoke CORBA services running on UNIX boxes, JavaScript clients can easily invoke code running on the mainframe, and Macintosh clients can start invoking Perl objects running on Linux. The list goes on. While some interoperability is achieved today through cross-platform bridges for specific technologies, once SOAP becomes standard, these linkage technologies will no longer be necessary.

4.2.2 SOAP -- Simple Object Access Protocol

Simple Object Access Protocol (SOAP), also referred, as Services Oriented Architecture Protocol (SOAP) is a network-neutral, lightweight protocol for exchanging of information in a decentralized, distributed environment [17]. SOAP is

considered as one of the most important standards being used in Web Services for its flexible invoking of methods on servers, services, components and objects in a platform independent manner. SOAP combines the existing practice of using XML and HTTP as a method invocation mechanism. SOAP is a protocol that acts as glue between heterogeneous software components. If developer can agree on HTTP and XML, SOAP offers a mechanism for bridging competing technologies in a standard way. The main goal of SOAP is to facilitate interoperability, so that it is widely viewed as the backbone to a new generation of cross-platform cross-language distributed computing architecture of Web Services. SOAP is not just an alternative to HTTP and XML. It can potentially be used in combination with a variety of other packaging and protocol schemes besides HTTP: MIME packages to support attachments, SMTP for scalable asynchronous messaging without the need for special middleware and many others (e.g., MQSeries, MSMQ, JMS).[18]

SOAP consists of three parts: first, an envelope that defines a framework for describing what is in a message and how to process it; second, a set of encoding rules for expressing instances of application-defined data types; and third a convention for representing remote procedure calls and responses. SOAP supports two communication styles: one is Remote Procedure Call (RPC) -synchronous invocation of operation returning a result, conceptually similar to other RPCs. This style is exploited by many Web Service tools and featured in many tutorials (e.g., [16]). Another is message-oriented style. This style provides a lower layer of abstraction, and requires more programming work.

Two concepts -- encoding and mapping -- in SOAP implementation need to be distinguished before moving to the next standard. In distributed computing environments, encoding defines how data values defined in application can be translated to and from protocol format. These translation steps are referred as serialization and deseralization, or, synonymously, marshalling and unmarshalling. For example, the protocol format for Web Services is XML, and service provider and requester are developed in Java. Thus, SOAP encoding tells the SOAP runtime environment how to translate from data structures constructed in Java into SOAP XML and vice versa. A mapping defines a ternary association between a qualified XML element name, a Java class name, and one of the encoding as introduced above.

A mapping specifies how, under the given encoding, an incoming XML element with a fully qualified name is to be converted to a Java class and vice versa.

Considerable progress is made in SOAP development. SOAP specification has been submitted to the World Wide Web Consortium (W3C) to propose the formation of a working group in the area of XML-based protocols. SOAP 1.1, 1.2 together with SOAP4J (SOAP for Java API) is available from Apache [19]. The third generation of Apache SOAP (essentially Apache SOAP 3.0) Axis [20] is designed as open source. The intention is to create a more modular, more flexible, and higher-performing SOAP implementation, with speed, flexibility, component-oriented deployment and transport framework as its new features.

4.2.3 UDDI – Universal Discovery Description and Integration

UDDI stands for Universal Discovery Description and Integration, which provides a mechanism to find available Web Services. UDDI creates a global, platform-independent, open framework to enable services provider and requester to discover each other, defines how they interact over the Web and share information in a global registry [21]. UDDI is a technical discovery layer, it defines: the structure for a registry of service providers and services; an API that can be used to access registries with this structure; and an organization and project defining this registry structure and its API. UDDI offers both a web-based user interface and a programmatic interface.
The information provided in a UDDI registration consists of three components:

• White page, including address, contact, and known identifiers.

• Yellow page, including industrial categorizations based on standard taxonomies.

• Green page, for the technical information about services that are exposed by the business.

There are four primary data types in a UDDI registry: businessEntity, businessService, bindingTemplate, and tModel. Figure 3 shows the relationship between all of these data types.

•**BusinessEntity** are the “white and yellow pages” of the registry. It provides business information about a service provider such as business name, contacts, description, identifiers and categories.

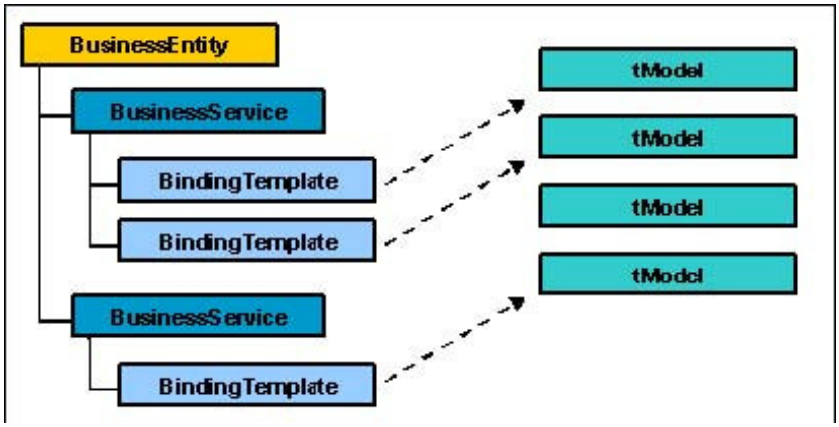


Figure 4.3: The UDDI Registry Structure[21]

•**BusinessService** are the "green pages" of the registry, which provides non technical service information. This information includes the address to make contact with a Web Service, as well as support for option information that can be used to describe both hosted services and services that require additional values prior to invoking a service.

•**BindingTemplate** are the "green pages" of the registry, which contains service access information. It points to a service implementation description (e.g. via a URL) in WSDL. This entity sometimes also called access locator.

• **tModel** is a technical fingerprint, holding metadata about type specifications as well as categorization information. Its attributes are key, name, optional description, URL. It does not contain detailed specifications about business services. It points to other sources that contain the service specifications. A tModel may point to a WSDL service interface description.

The use of UDDI has been described in six ways, UDDI operator node, E-marketplace UDDI, Portal UDDI, Partnership catalog UDDI, Internal enterprise application integration UDDI and Test bed UDDI. Currently, the most important form of UDDI is the public UDDI [22]. There are three public UDDI registries available at this moment , these registry nodes are operated by IBM, HP and Microsoft, in which registration is free [21]. The operators replicate the registrations across all nodes on a regular basis thus resulting in a complete set of registered records available to all. The operators all support a common set of SOAP APIs that will ensure the integrity and availability of the information provided.

UDDI is currently supported by more than 310 companies, but not a standards body

yet. Future features will address the ability to locate products and services, define Web Services implementation conventions and provide the ability to manage hierarchical business organizations, communities and trade groups. The driving goal is to provide a public specification for Web Services interoperability.

4.2.4 WSDL – Web Services Description Language

The Web Services Description Language (WSDL) is an XML-based interface definition language that provides a way to catalog and describe Web Services [23]. It is used to automate the details involved in applications communication. WSDL defines Web Services interface, Web Services access protocol (e.g. SOAP over HTTP) and Web Services contact endpoint (namely, Web Services URL). Compliant server applications support these interfaces, and client users can learn from the document how a service should be accessed.

WSDL documents are divided into two types: *service interfaces* and *service implementations* (Figure 4.4)[23], which are used in publishing and finding WSDL service descriptions in a UDDI Registry

A *service interface* is described by a WSDL document that contains the *types*, *import*, *message*, *portType*, and *binding* elements. A service interface contains the WSDL service definition that will be used to implement one or more services. It is an abstract definition of a Web Service, and is used to describe a specific type of service. A service interface document can reference another service interface document using an import element [22].

The WSDL service implementation document will contain the *import* and *service* elements. A service implementation document contains a description of a service that implements a service interface. At least one of the import elements will contain a reference to the WSDL service interface document. A service implementation document can contain references to more than one service interface document. The *import* element in a WSDL service implementation document contains two attributes. The *namespace* attribute value is a URL that matches the *targetNamespace* in the service interface document. The *location* attribute is a URL that is used to reference the WSDL document that contains the complete service interface definition. The *binding* attribute on the *port* element contains a reference to a specific binding in the service interface document.

The service interface document is developed and published by the *service interface provider*. The service implementation document is created and published by the *service provider*. The roles of the service interface provider and service provider are logically separate, but they can be the same business entity.

Publishing and finding WSDL descriptions is explained and the mapping from WSDL to UDDI is examined (Figure 4.5). A complete WSDL service description is a combination of a service interface and a service implementation document.

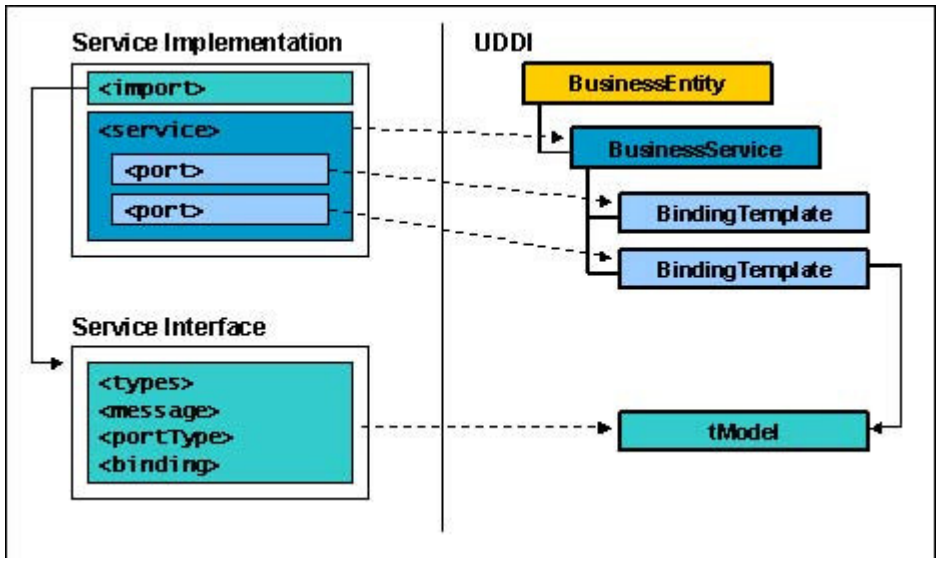


Figure 4.4: An Overview of the mapping from WSDL to UDDI [22]

Since the service interface represents a reusable definition of a service, it is published in a UDDI registry as a *tModel*. The service implementation describes instances of a service. Each instance is defined using a WSDL service element. Each *service* element in a service implementation document is used to publish a UDDI *businessService*.

When publishing a WSDL service description, a service interface must be published as a *tModel* before a service implementation is published as a *businessService*.

The development of WSDL is fast growing, the specification (WSDL 1.1 Specification [23] and WSDL4J [61]), supported tooling (WSTK Toolkit [24], WSAD [25], WSDE [26], (will be explained in detail in Chapter 3), etc.) and articles and tutorials are all widely available.

4.2.5 Conceptual Architecture Stack

Based on these emerging technologies, industry leaders are proposing more standards and specifications in the related area. Such as IBM proposed Web Services Flow Language (WSFL), which enables the same type of seamless integration across processes and transaction life cycles that make use of many Web Services [27]. Although existing standards and ongoing work are rapidly progressing, there are not yet any finished standards, as stated, SOAP and WSDL specifications have been proposed to the W3C.

Other issues are currently investigated, such as Web Services management, QoS (Quality of Service), security and performance, where security addresses a big concern at this moment [28]. In order to have an insight of network-accessible Web Services, Web Services conceptual "Stack" can be seen as a good example to illustrate these standards as different layers (Figure 4.6) [28]

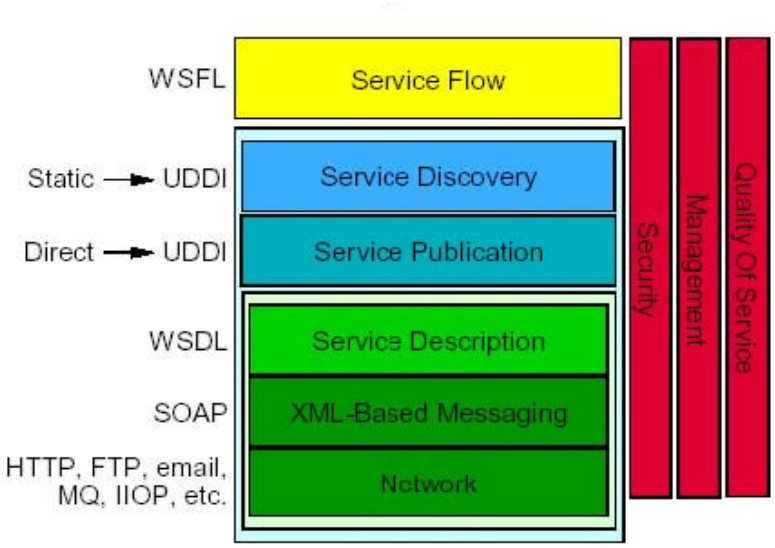


Figure 4.5: Conceptual Architecture Stack [28]

Without going into much detail of the component of the stack, the intention here is to show that different *de facto* standards for comprising the Web Services architecture (XML, SOAP, WSDL, UDDI) complement each other nicely without depending unnecessarily on each other, but complete Web Services concept as a whole.

4.2.6 A Typical Web Service Invocation

1. As we said before, a client may have no knowledge of what Web Service it is going to invoke. So, our first step will be to *discover* a Web Service that meets our requirements. For example, we might be interested in locating a public Web Service which can give me the weather forecast in US cities. We'll do this by contacting a *discovery service* (which is itself a Web service).

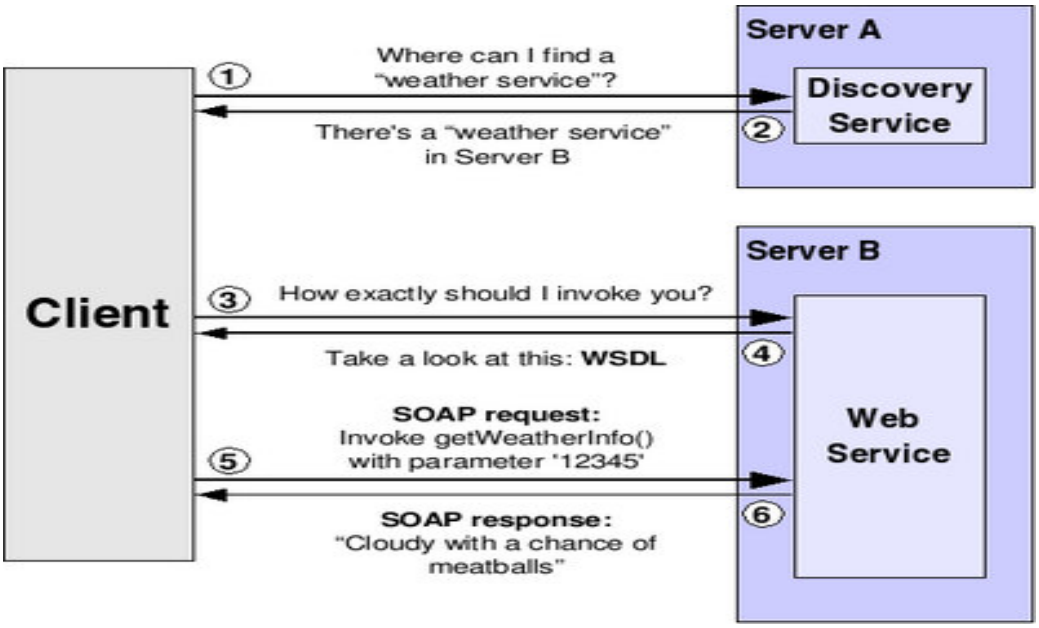


Figure 4.6: A Typical Web Service Invocation [61]

2. The discovery service will reply, telling us what servers can provide us the service we require.
3. We now know the location of a Web Service, but we have no idea of how to actually invoke it. Sure, we know it can give me the forecast for a US city, but how do we perform the actual service invocation? The method I have to invoke might be called "**string getCityForecast(int CityPostalCode)**", but it could also be called "**string getUSCityWeather(string cityName, bool isFarenheit)**". We have to ask the Web Service to *describe* itself (i.e. tell us how exactly we should invoke it)
4. The Web Service replies in a language called WSDL.

5. We finally know where the Web Service is located and how to invoke it. The invocation itself is done in a language called SOAP. Therefore, we will first send a *SOAP request* asking for the weather forecast of a certain city.
6. The Web Service will kindly reply with a *SOAP response* which includes the forecast we asked for, or maybe an error message if our SOAP request was incorrect.

4.3 Comparison with Other Approaches

Recent efforts in peer-to-peer (P2P) computing and enterprise computing address problems and issues, which are equally valid in a Web Services environment. While Web Services technologies are currently distinct from other major technology trends, such as Internet, distributed object middleware, Grid computing, and peer-to-peer computing, Web Services technologies can benefit significantly from growing into the problem space addressed by these other trends, and certainly the other way applies as well.

4.3.1 Peer-to-Peer Computing

Collaboration interoperating with each other on the Web Services are essentially peers interacting in a P2P network as discussed in the earlier communication evolution final stage. Peer-to-peer computing includes two kinds of sharing, first, as implemented in Internet communication and file/data sharing tools like Napster [29], Gnutella [30], and Freenet [31]; and second, as Internet computing as implemented by CPU sharing systems such as SETI@home (Search for Extraterrestrial Intelligence) [32], Parabon [33], and Entropia [34], are examples of more general sharing modalities and computational structures beyond traditional client-server systems. These types of distributed computing implementation can be interpreted as: harness the unused capacity of Internet-connected computers and make them available for working on computational intensive problems, which would otherwise require a super computer or workstation/server cluster to solve. The application area of these initiatives contains life science and financial services, which add the value, as maximizing ROI (Return On Investment) and utilizing the corporate PCs, accelerating research to super computer speed, dramatically shortening the time-to-discovery and accelerate critical analyses.

These features characterize virtual organizations where information and resource sharing can take place among any subset of participants. These technologies and systems present a radical paradigm shift from client-server systems. These systems have so far focused entirely on vertically integrated solutions, rather than seeking to define common protocols that would allow for a shared infrastructure and interoperability, which Web Services are trying to achieve. Also the forms of sharing targeted by various applications are quite limited; for example, file sharing with no access control, computational sharing with a centralized server or multi-user text based collaboration.

JXTA Technology [35] contains several open, generalized peer-to-peer protocols that allow any connected device on the network from cell phone to PDA from PC to server to communicate and collaborate in a peer-to-peer manner. The objective of the project is to build interoperable, platform independent and ubiquitous peer-to-peer computing. It differs from just mentioned peer-to-peer implementations and similar to the Web Services technology in its aiming at highest abstraction of a set of protocols use XML-encoded messages and can be implemented on top of TCP/IP, HTTP, Bluetooth and many other protocols. The similarity also exists in its being away from APIs and being programming independent. JXTA technology is still evolving and will soon be open-sourced and co-developed by many contributors [35]. Many of the protocols and the objectives defined by JXTA, are equally valid and significant in Web Services environments. These categories pretty much summarize the current development and use of peer-to-peer technology and examine the difference and similarity to Web Services.

At its core, P2P computing is a kind of service-oriented architecture in that it enables distributed computing through the loose coupling of systems, with emphasis on resource (instead of service) description and dynamic discovery. The major differences are not technical, but in terms of maturity. Although Web Services standards are maturing and their interoperability is being tested in open environments, P2P standards have been very slow to emerge, hindering their acceptance as a valid option for e-business. Eventually, however, both Web Services and P2P standards will clearly converge, especially when it comes to service description, dynamic discovery and security.

4.3.2 Object-Middleware

The justification of this new Web Services distributed computing model is cross-platform and cross-programming language interoperability. These features are pretty much in the heart of Object Middleware concept, which is gaining wide acceptance as a generic software infrastructure for distributed computing systems. A large number of applications are designed and implemented as a set of collaborating objects using object middleware, such as CORBA [2, 36], Java/RMI [37] and DCOM [38]. Object middleware offers interaction support to application objects, which may be deployed in different computer nodes with a transparent manner to developers. It provides a uniform interaction pattern, independent of the underlying node and network technologies. As the term services characterized in Web Services context, from an abstract point of view, an object is an identifiable, encapsulated entity that provides services to other objects through its interface. The interface of an object shields the other objects from the internal characteristics of this object, like its internal data representation, algorithms or executable code.

Web Services are often compared to classical remote procedure calls (RPCs) and distributed computing frameworks such as CORBA, DCOM, and J2EE. However, Web Services extend the concept of being another distributed computing or RPC mechanism. Web Services provide universal access; CORBA, DCOM and J2EE are implementation alternatives. As following, the shortcomings of previous solutions are analyzed and future requirements for successful solutions are proposed:

•**JAVA/RMI** distributed method resolution mechanism in a Java-to-Java environment; integration is done at the binary level and not at a descriptive level over a multi-platform carrier mechanism. It can be used as an endpoint integration mechanism beyond Web Services. Or could be used as a carrier of messages in implementation. It has no bearing on the Web Services application developer and no lookup function available.

•**DCOM/COM+** a distributed software component model and in some respects similar to CORBA but of course Microsoft's strategic middleware architecture for Windows. This is also a tight server/server or client/server integration architecture, without effective integration behavior or any descriptive abstraction support.

•**CORBA/IIOP** shortened as Common Object Request Broker Architecture (CORBA) relies on a protocol called the Internet Inter-ORB Protocol (IIOP), is the

Object Management Group's (OMG) specification for achieving interoperability between distributed computing nodes. The objective is to define an architecture that would allow heterogeneous environments to communicate at the object level regardless of who designed the two endpoints for the distributive application. A CORBA object is represented to the outside world by an interface with a set of methods. The interaction between a client process and an object server are implemented as object-oriented RPC-style communications. Although CORBA has been implemented on various platforms, the reality is that any solution built on these protocols will be dependent on a single vendor's implementation. Thus, if one were to develop a CORBA application, all participating nodes in the distributed application would have to be running the same ORB product. Now there are cases where CORBA ORBs from different vendors do interoperate. However, that interoperability does not extend into high-level services such as security and transaction management.

Technology	CORBA	DCOM	Web Services
Naming of communication endpoints	Interoperable object reference (IOR)	OBJREF	URL/URN
Underlying remoting protocol	Internet Inter-ORB Protocol (IIOP)	Object Remote Procedure Call (ORPC)	SOAP
Interfaces	Multiple	Multiple	Multiple – WSDL
Payload type	Binary	Binary	Text message
Payload parameter value format	Common Data Representation (CDR)	Network Data Representation (DR)	XML
Server Address Resolution	Naming service	(Directory)	IP routing, URN
Message dispatch	ID based	ID based	Namespace and parameter types
Client/Server coupling	Tight	Tight	Loose

Table 4.1: Differences between distributed computing technologies

Table 4.1 gives an overview of some important differences between Web Services

and the other technologies. (J2EE uses CORBA IIOP as transport protocol, so there is no J2EE column in the table.)

The architectures of Java/RMI, DCOM and CORBA provide mechanisms for transparent invocation and accessing of remote distributed objects. Though the mechanisms that they employ to achieve remoting may be different, the approach each of them takes is more or less similar. Web Services programming model is not viewed as a reappearance or revitalization of CORBA or the others. Instead, it is seen as a new open solution that addresses the same distributed computing issue that CORBA addressed with the further objective of improving on some of CORBA's shortcomings. Of course, CORBA also has many of the features Web Services support and many other more.

The technology of Web Services offers a new programming model for building distributed applications using open Internet standards. This new distributed computing solution exploits the openness of specific Internet technologies to address many of the interoperability issues of CORBA.

4.4 Web Services and Grid Computing

The nature of Web Services has the same properties as Grid computing: interoperability, protocol architecture, standard-based open architecture and standard services, application programming interface and software development kits.

Similarity of Web Services and Grid computing also reflect in their layered architectures. Grid computing has fabric, connectivity, resource, collective, application layers, which can be mapped into different layer in the Internet Protocol Architecture. Similar

One of the significant properties of Web Services is its composable, which can be exposed through WSDL. Therefore all those composition mechanisms are really interchangeable by means of WSDL. This is a layer of abstraction that adds up to the robustness of Web Services technology, which can very much contribute to the Grid computing.

The major values Web Services can bring to Grid computing can be summarized as:

Enable interoperability:

- Definition of service interface

- Identification of the protocol used to invoke a particular interface

Encapsulation:

- Realize local/remote transparency
- Separate and distinct the common interface and diverse implementation

Composition:

- Individual service form sophisticated services
- Component-based workflow

Uniform services semantics:

- Transient service instance emphasize on lightweight entities/short-lived activities
- Standard interfaces (discovery, dynamic service creation, lifetime management, notification, etc.)

Integration is easier if service functions can be expressed in a standard form, so that any implementation of a service is invoked in the same manner. With respect to this, WSDL is thought to be most valuable standard for Grid services combination for its composable, interface and implementation separation, which generates a model for Services with ports communicating by message with a general XML specified structure.

A critical requirement in a distributed, multi-organizational Grid environment is how to enable interoperability, which can be interpreted as two parts: one is the definition of service interface and the other one is identification of the protocol that can be used to invoke a particular interface.

Chapter 5
Problem Formulation

As seen in previous chapters the Grid community has not yet been able to fully resolve the issue of Resource Discovery due to various technical constraints and geographical limitations such as autonomous, heterogeneous resources, dynamic nature and status of resources, geographical dispersion of resources, large number of users and large distributed networks, different operating systems/platforms, different administrative domains, lack of portability, availability status of resources and different technology policies. It is not easy to track or locate the right resource in a vast interconnected environment. Therefore, Techniques adopted for resource discovery should be both location and platform independent.

Resource discovery on Grids is quite similar to the discovery of resources on the Web, since in both cases the user is transparent of the specific resource location. For this purpose, a consistent and controlled relationship among the various resources defined on a Grid is needed.

The main thrust is on the development of a framework that allows Grid resource discovery using the underlying web service as vehicles or transporters of these services applications to be installed and used as Web services accessed through a Grid portal.

We focus on two specific issues:

- How can a Grid resource discovery be captured as a Web service?
- Using .Net Framework to address Windows-based Grid computing.

  Software to enable Grid computing has been primarily written for Unix-class operating systems, thus severely limiting the ability to effectively utilize the computing resources of the vast majority of desktop computers i.e. those running variants of the Microsoft Windows operating system.

Chapter 6
Design and Implementation

There is rapidly emerging interest in Grid computing from commercial enterprises. A Microsoft Windows-based Grid computing infrastructure will play a critical role in the industry-wide adoption of Grids [9][14] [17][22] due to the large-scale deployment of Windows within enterprises. This enables the harnessing of the unused computational power of desktop PCs and workstations to create a virtual supercomputing resource at a fraction of the cost of traditional supercomputers. However, there is a distinct lack of service-oriented architecture-based Grid computing software in this space. To overcome this limitation, we have developed a Windows-based Grid resource discovery framework implemented on the Microsoft .NET Platform.

While the notion of Grid computing is simple enough, the practical realization of Grids poses a number of challenges. Key issues that need to be dealt with are security, heterogeneity, reliability, application composition, scheduling, and resource management [13]. The Microsoft .NET Framework [3] provides a powerful toolset that can be leveraged for all of these, in particular support for remote execution (via .NET Remoting [4] and web services [23]), multithreading, security, asynchronous programming, disconnected data access, managed execution and cross-language development, making it an ideal platform for Grid computing middleware.

6.1 System Architecture and Data Flow

The architecture of TU Grid Resource Discovery System is as shown in the figure 6.1. This system facilitates resource information publication, management and browsing. It allows resource providers and consumers to use a web browser as a simple graphical client to access the web service.

Resource information retrieval module enables application (e.g. resource broker) to query to search the status database and find a suitable resource to meet the job execution requirement.

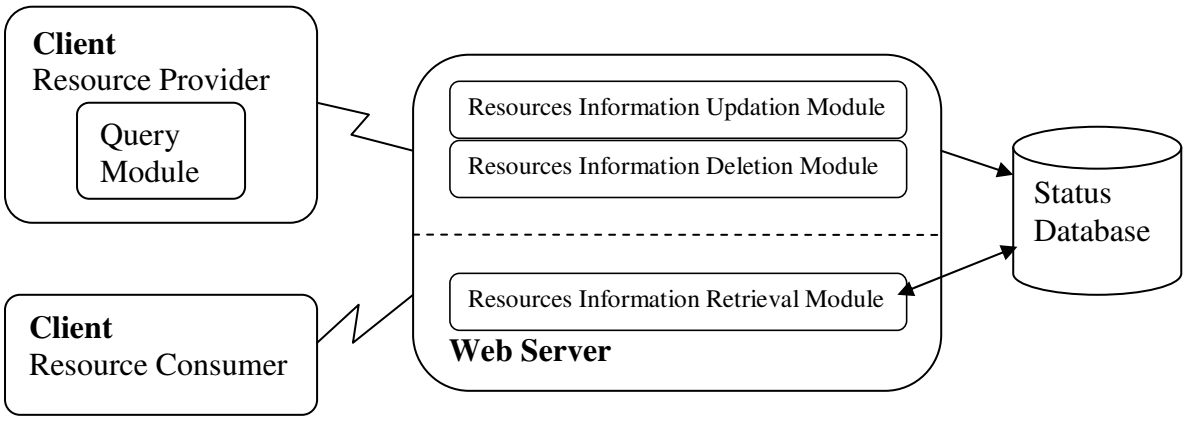


Figure 6.1: System Architecture

Resource information Updation module and Resource information deletion module allow providers to publish its resource information in a non-dedicated fashion to have control over the time for which resources are used by the application and delete this information from the status database when the provider desires.

Both the components receive client requests through a HTTP server. Additionally a database is configured for recording the information of resources and resource providers. Screenshot 6.11 indicates the addition of web reference to consume web service in a web application.

The proposed system described above will implement resource information publication, management and browsing as shown in figure 6.2.

6.2 Resource Provider

The Resource Provider is built using SOAP (Simple Object Access Protocol). The main benefit from using SOAP is multi-language supporting. Since there is support for the SOAP protocol in many programming language, for example, C++, Java and VB, application developers have flexibility of selecting the programming environment of their choice. In addition to the standard SOAP client infrastructure, the system clients need XML message building and parsing capability depending on the query and response specification.

6.2.1 Architecture and Data Flow

Windows Service captures resources and resource provider information from system registry. Web application gets this information from windows service and passes to web server according to provider's desire. By clicking on Refresh button information

is fetched from windows service. By pressing *Update Resource Information* button information (Screenshot 6.5) is updated in status database by calling Insert () web method at web server. By pressing *Delete Resource Information* button (Screenshot 6.5) information is deleted in status database by calling delete () web method at web server.

Figure 6.3 below presents the architecture of the Resource Provider client APIs and their interaction with the web server.

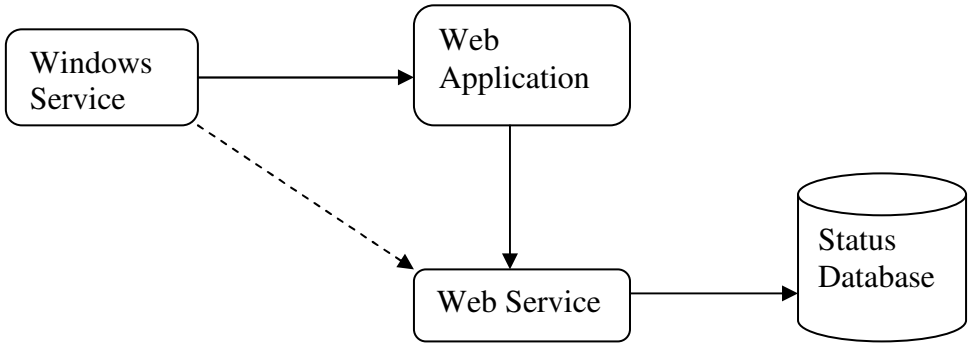


Figure 6.3: Resource Provider Architecture

Figure 6.4 gives the information flow diagram of resource provider.

The basic SOAP invocation methods supported by the provider are:

(1) Insert () – Publishes system and providers information. This Includes total physical memory, available physical memory, total virtual memory, available virtual memory, system operating system, system user, and system domain and system machine. WSDL (Web Services Description Language) is generated when insert web method is invoked. (Screenshot 6.6)

(2) Delete () – This web method deletes system information to indicate provider has been shutdown or is not willing to share its information. System name is provider as parameter to this web method. WSDL is generated when Delete web method is called. (Screenshot 6.7)

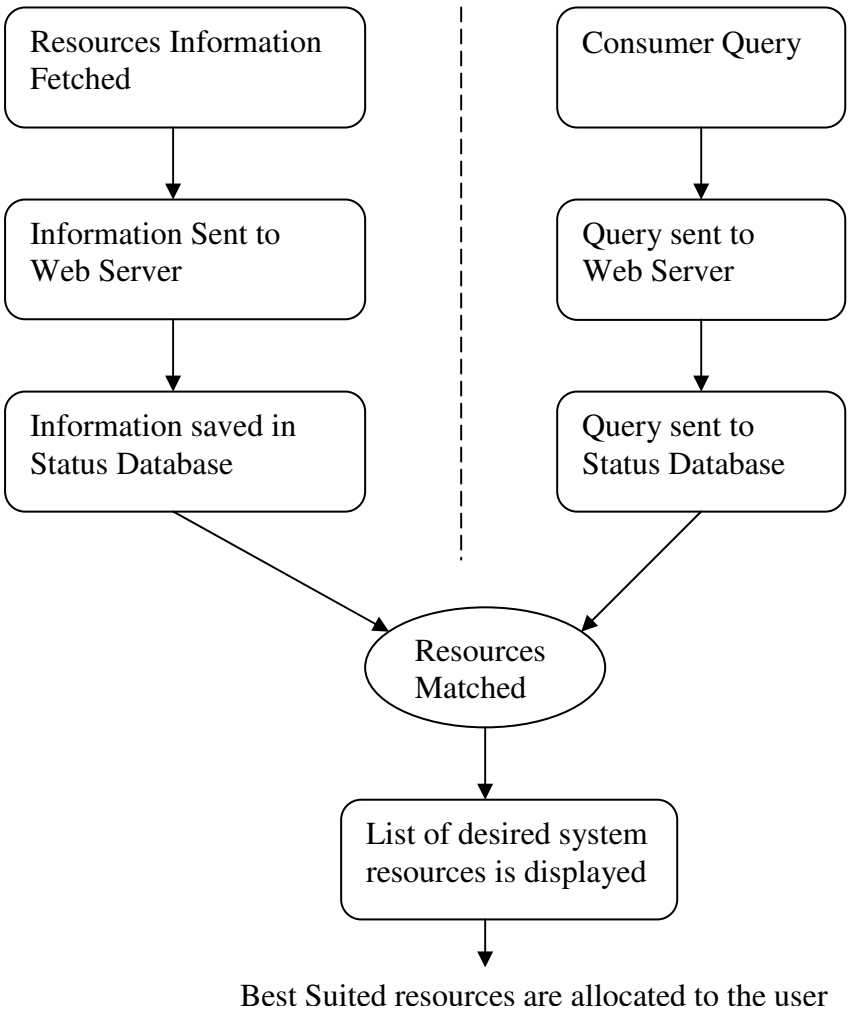
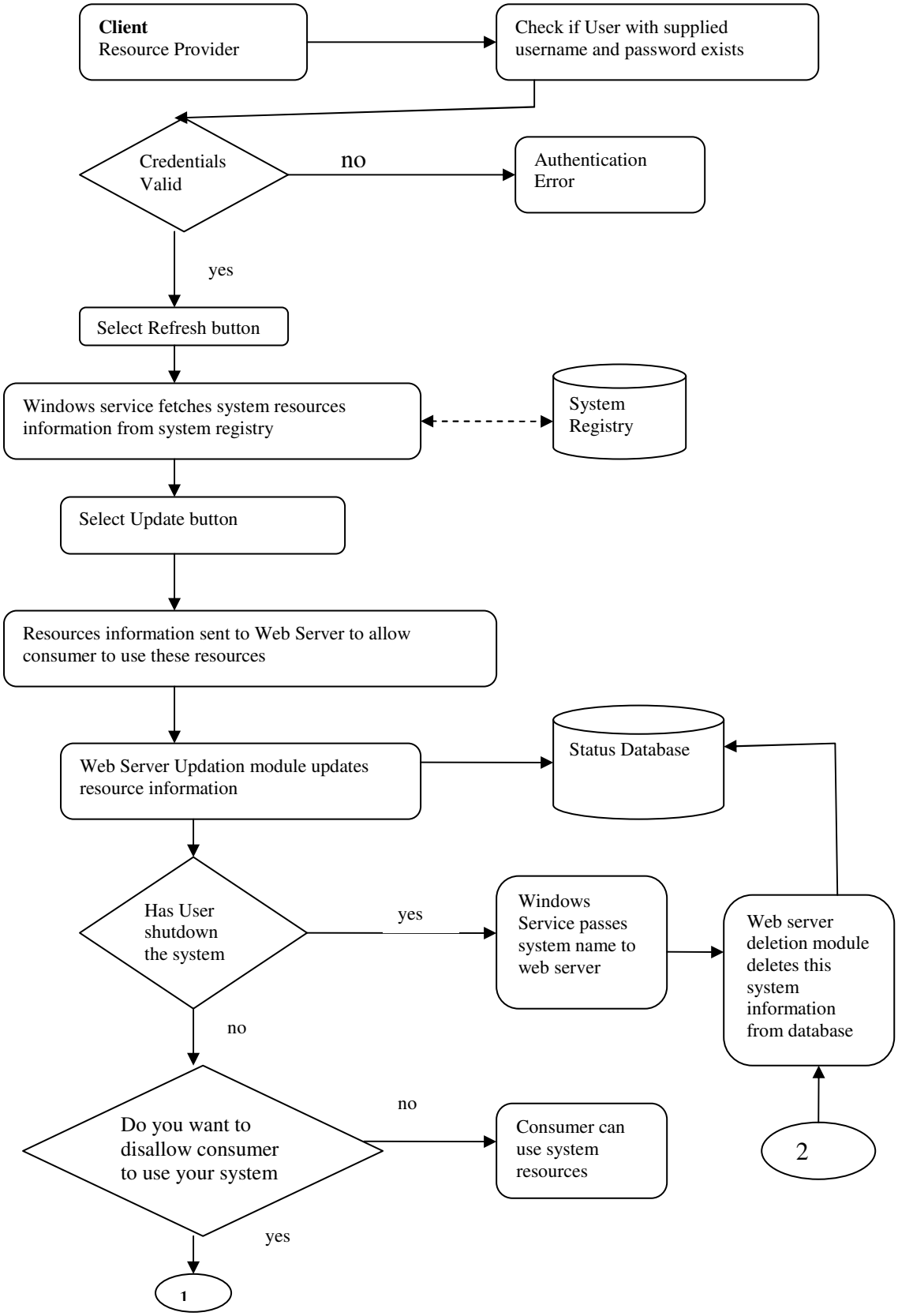


Figure 6.2: Information Flow Diagram: System


Client
Resource Provider
Check if User with supplied username and password exists
Credentials Valid
no
Authentication Error
yes
Select Refresh button
Windows service fetches system resources information from system registry
System Registry
Select Update button
Resources information sent to Web Server to allow consumer to use these resources
Web Server Updation module updates resource information
Status Database
Has User shutdown the system
yes
Windows Service passes system name to web server
Web server deletion module deletes this system information from database
no
Do you want to disallow consumer to use your system
no
Consumer can use system resources
2
yes
1

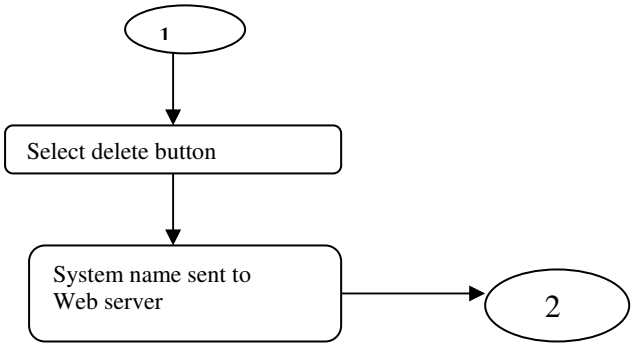


Figure 6.4: Information Flow Diagram: Resource Provider

6.2.2 Brief Overview of Windows Service

Windows Services are applications that operate in the background and are independent of the user who is logged in. The tasks carried out by Windows Services are typically long running and have little or no direct interaction with user (so they don't have user interface). They start when the computer is booted and continue to run until that is shutdown. For Example: Microsoft SQL Server, Internet Information Server, antivirus software all use windows services to perform tasks in response to events that occur on the system overall.

6.2.3 Characteristics of Windows Service

- A Windows Service can start before a user logs on. The system maintains a list of Windows Services and they can be set to start at boot time. They can also be installed so that they require a manual startup and will not start at boot.

- A Windows Service can run under a different account from that of the current user. Most Windows Services provide functionality that needs to be running all the time and some load before a user logs on, so they cannot depend on a user being logged on to run.

- A window service has its own process. It does not run in the process of a program communicating with it.

- A Windows Service typically has no user interface. This is because the service

may be running under a different account from that of the current user, or the service may start at boot time, which would mean that the calls to put up a user interface might fail because they are out of context.

- A Window Service requires a special installation procedure; just clicking on a compiled EXE won't run it. The program must be run in a special context in the operating system and a specific installation process is required to do the configuration necessary for a Windows Service to be run in this special context.

  Installation can be done by following command (Screenshot 6.1)

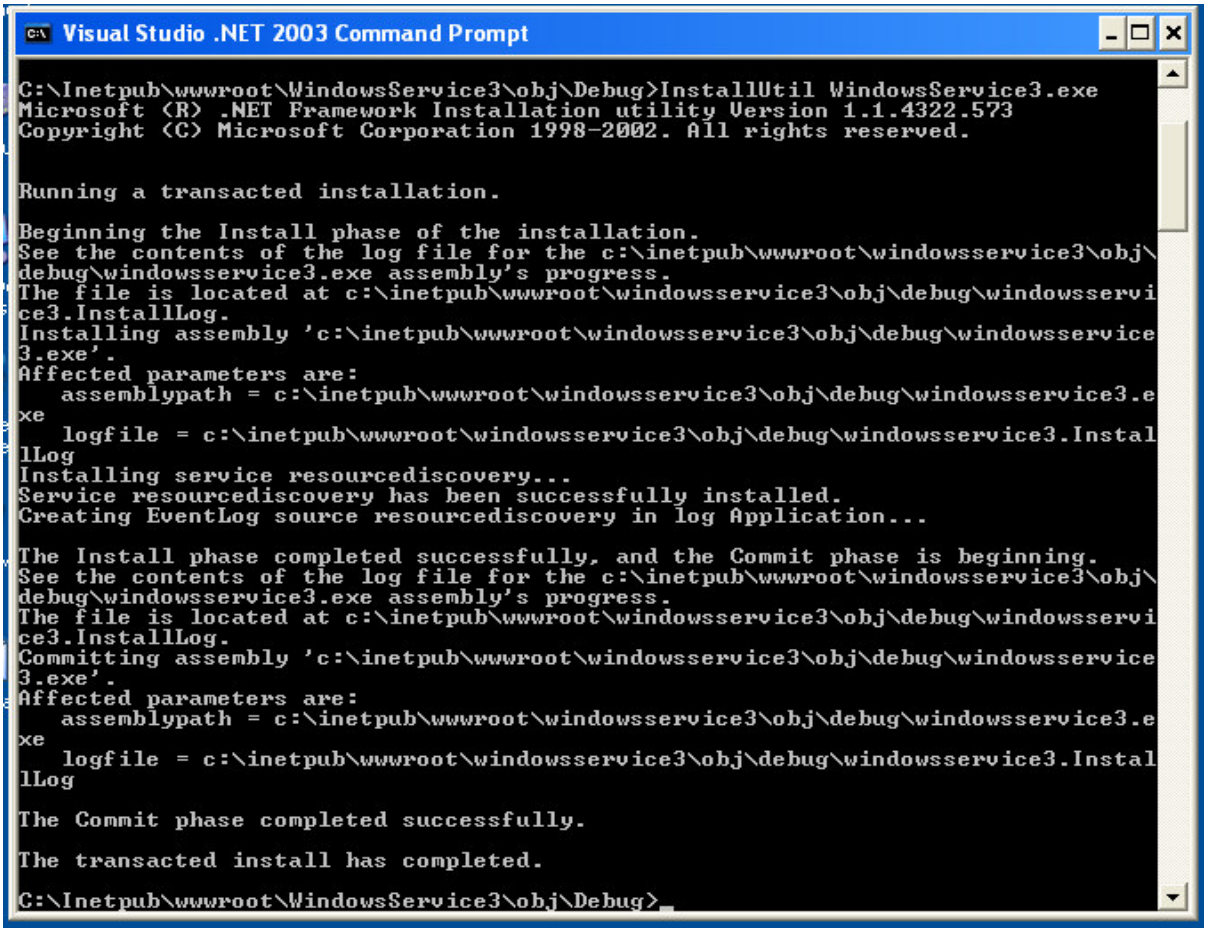
  InstallUtil WindowsService.exe

  Every time some modification to windows service is done, it should be uninstalled by following command (Screenshot 6.2)

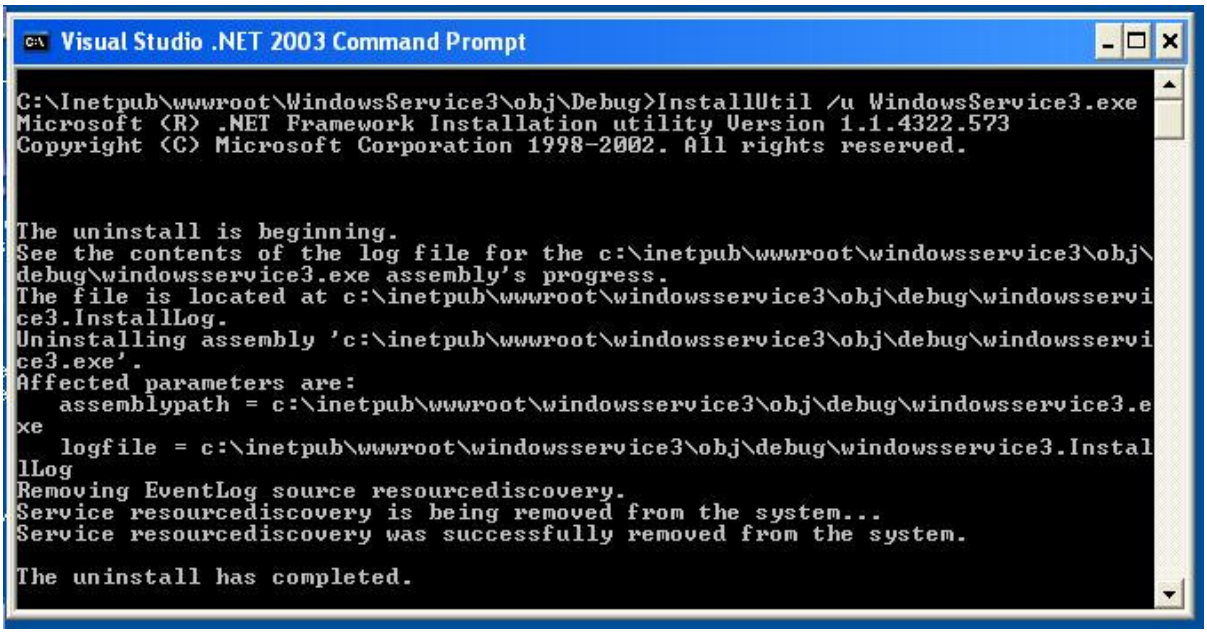
  InstallUtil /u WindowsService.exe

- A Window service works with Service Control Manager (Screenshot 6.3). The Service Control Manager is required to provide an interface to the Windows Service. External programs that want to communicate with a Windows Service must go through the Service Control Manager. The Service Control Manager is an operating system level program, but it has a user interface that can be used to start and stop services, and this interface can be accessed through the Computer Management section of the Control Panel.

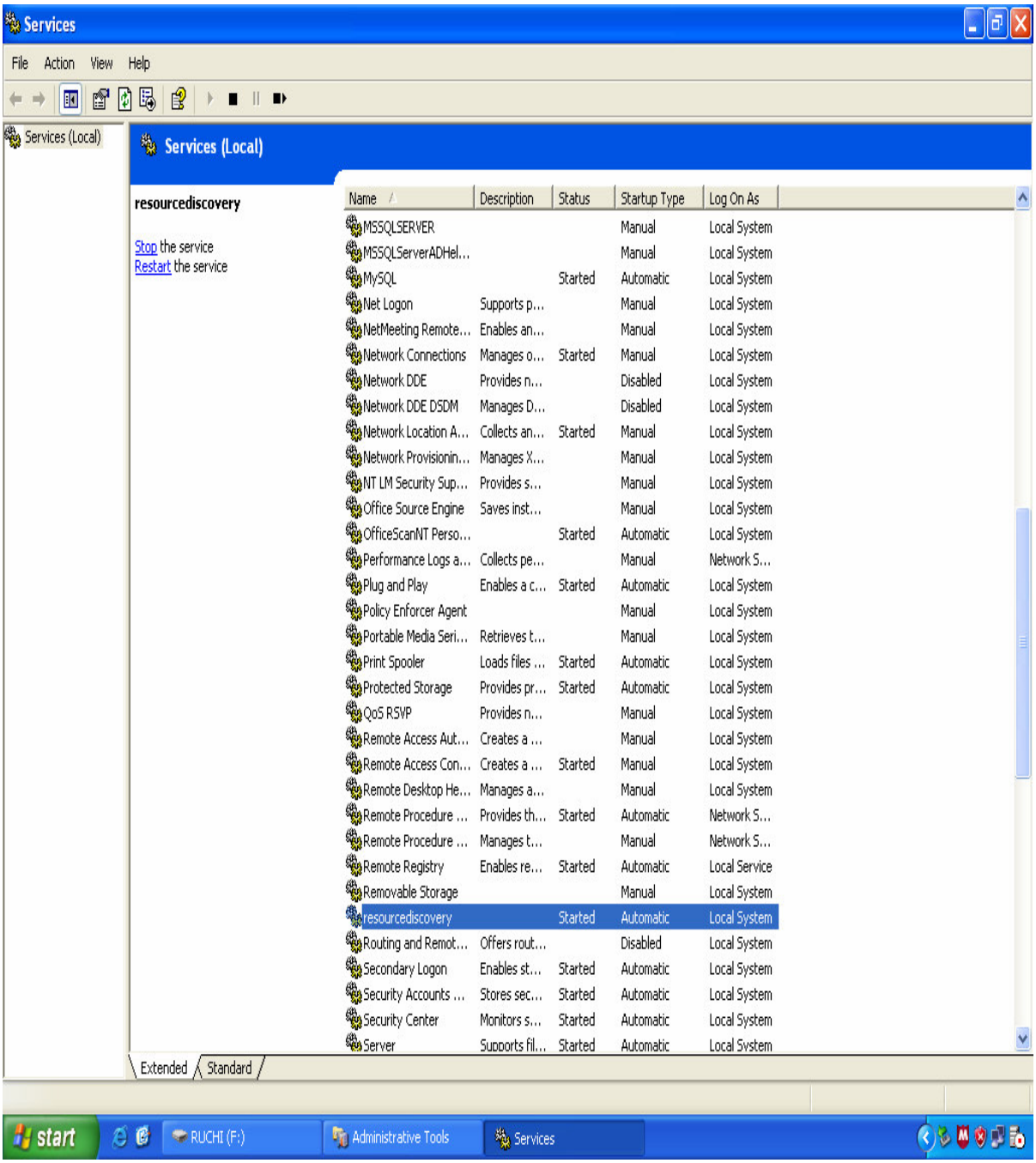
In addition, security issues are also addressed. A login authentication mechanism for identifying registered providers is employed in the provider administration. A database of authenticated users and their password is maintained at the server. When ever the provider logins this database is consulted to check validity. (Screenshot 6.4)



Screenshot 6.1: Window Service Installation

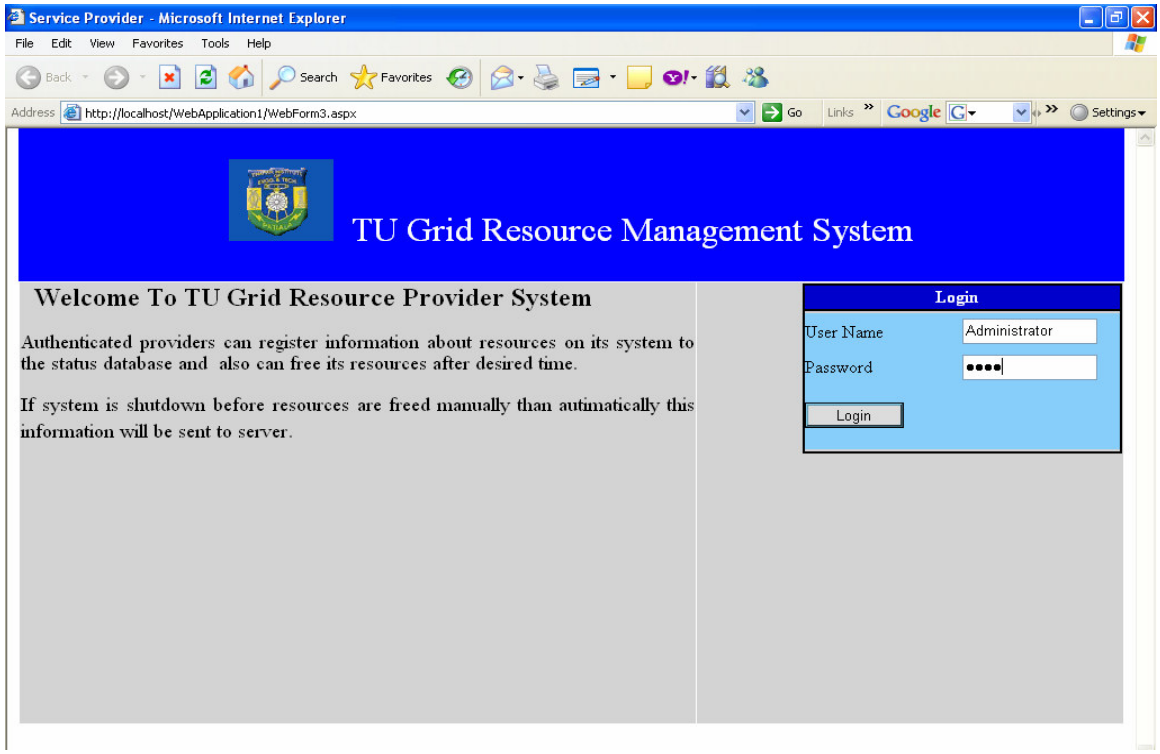


Screenshot 6.2: Windows Service Uninstallation

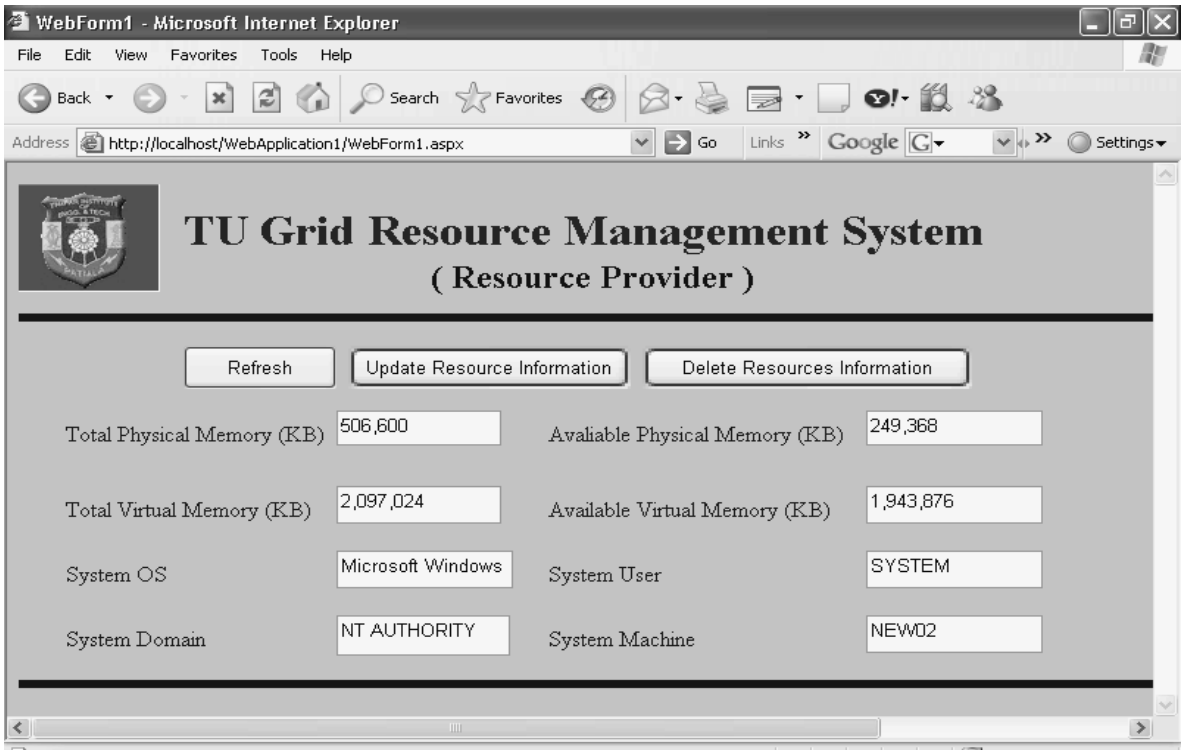


Screenshot 6.3 Service Control Manager: Resource Discovery service started

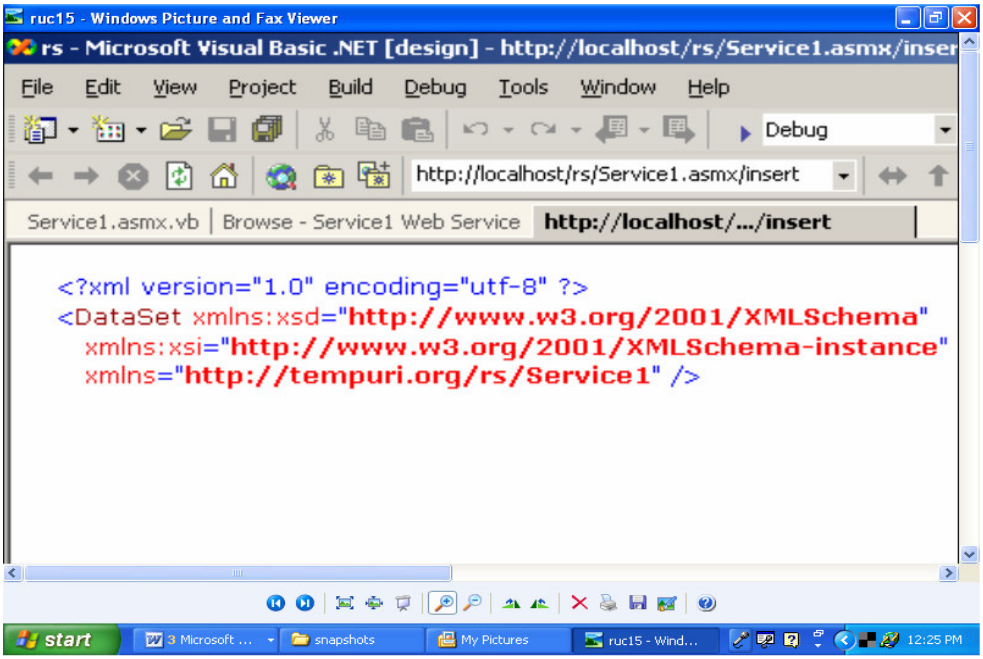
6.2.4 Experimental Setup



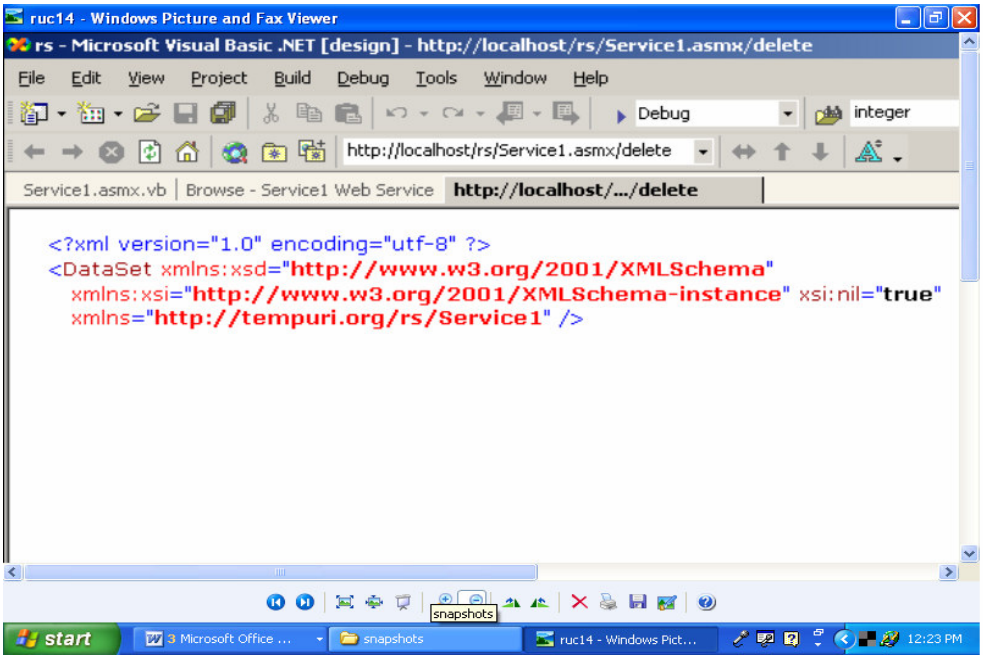
Screenshot 6.4: Authentication form of Resource Provider



Screenshot 6.5: Homepage of TU Grid Resource Provider System



Screenshot 6.6: WSDL for web method: insert ()



Screenshot 6.7: WSDL for web method: Delete ()

6.3 Resource Consumer

Authenticated Consumers can retrieve list of resources available on system connected to Grid from status database by specifying its requirements (Screenshot 6.8). If Users does not specify constraint on some resource than minimum value is taken as default value.

6.3.1 Architecture and Data Flow

Web application calls retrieve () web method at web server and passes requirements of consumer as parameters. This is achieved by pressing retrieve button on retrieval form of TU Grid Resource Consumer System. (Screenshot 6.9) Web Server invokes retrieve () method and fetches list of relevant resources from the status database and again gives to web application so as to show to the consumer.

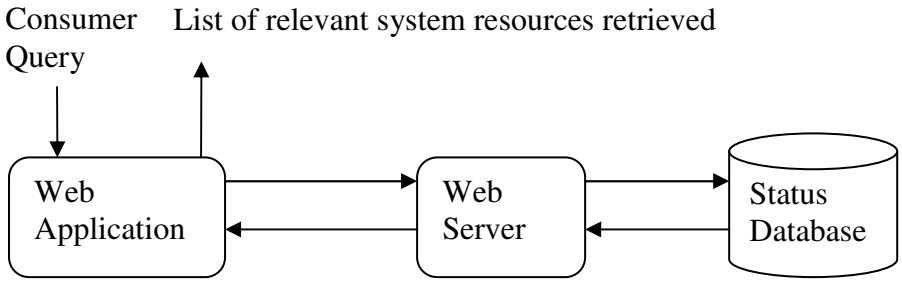


Figure 6.5: Resource Consumer Architecture

The basic SOAP invocation method supported by the consumer is:

(3) Retrieve ()- This web method fetches relevant resource list from the status database. Physical memory, virtual memory, system operating system are passed as parameter to the web method on the basis of which result is displayed. WSDL is generated when retrieve web method is called. (Screenshot 6.10)

Figure 6.6 explains the information flow for resource consumer.

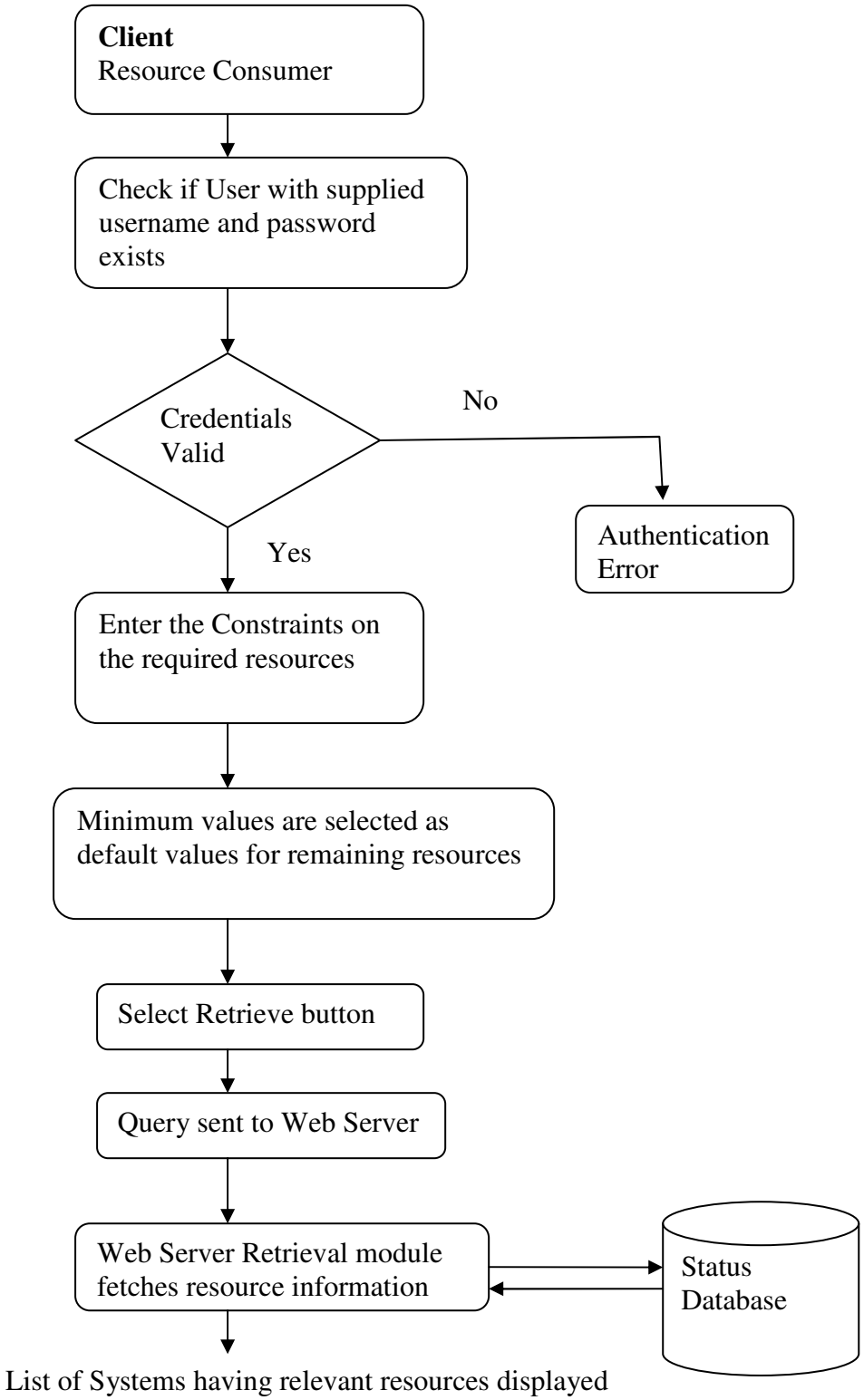
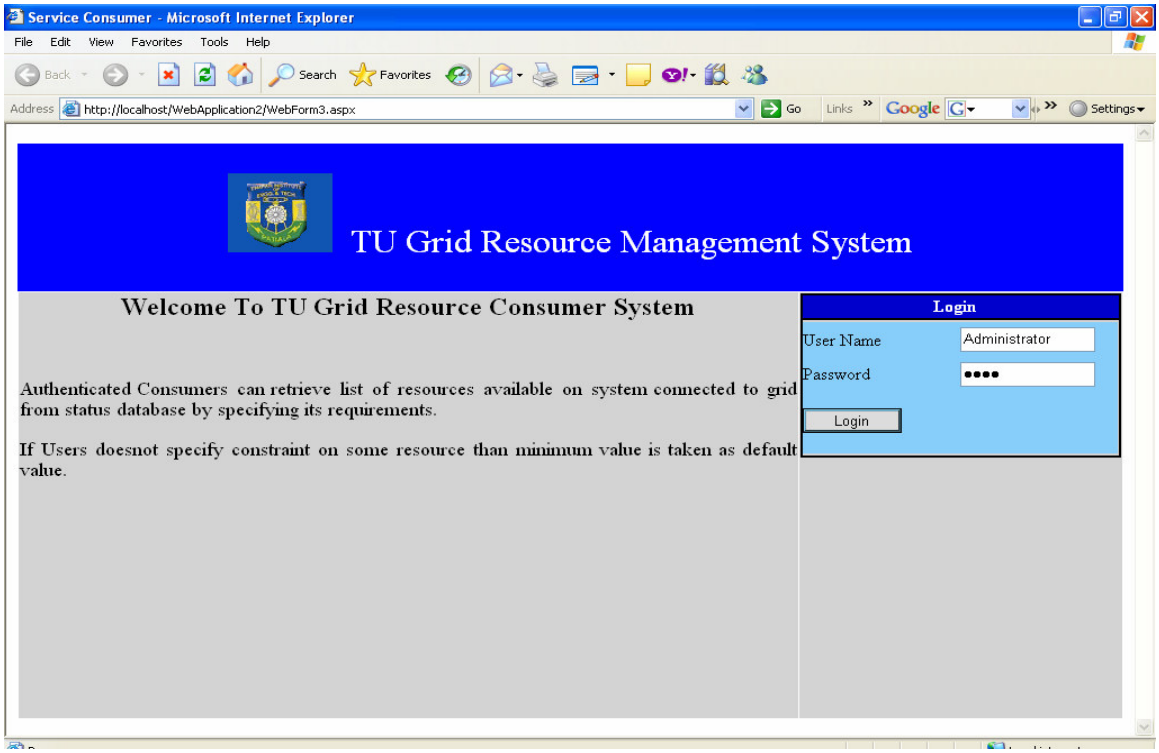
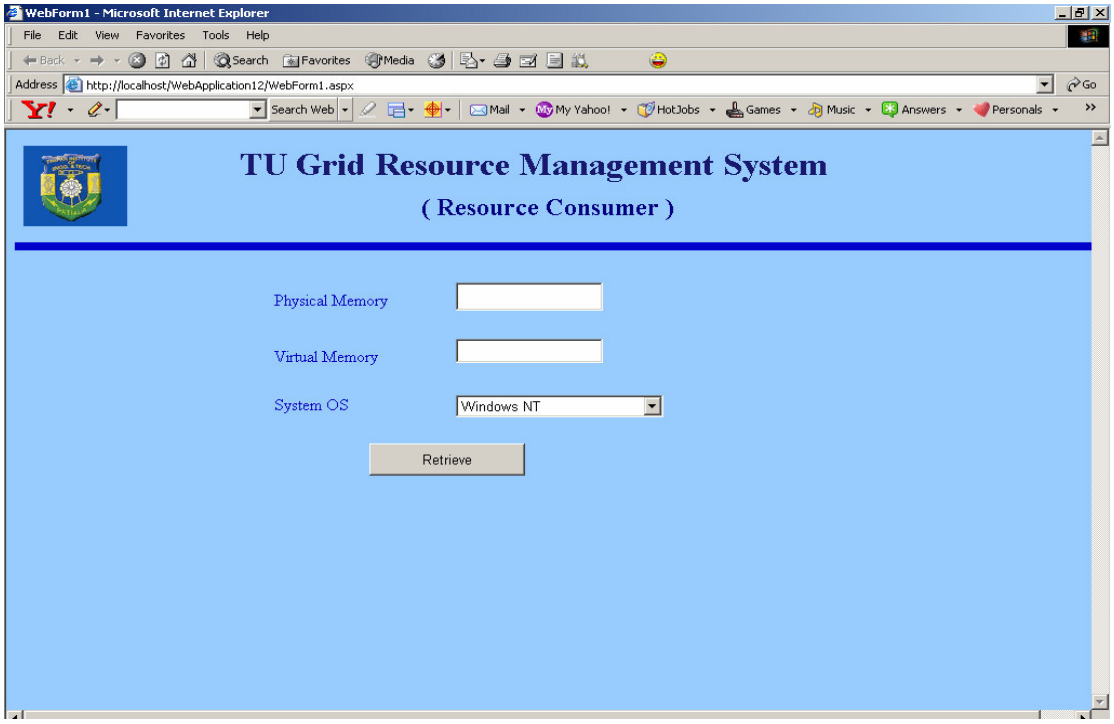


Figure 6.6: Information Flow Diagram: Resource Consume

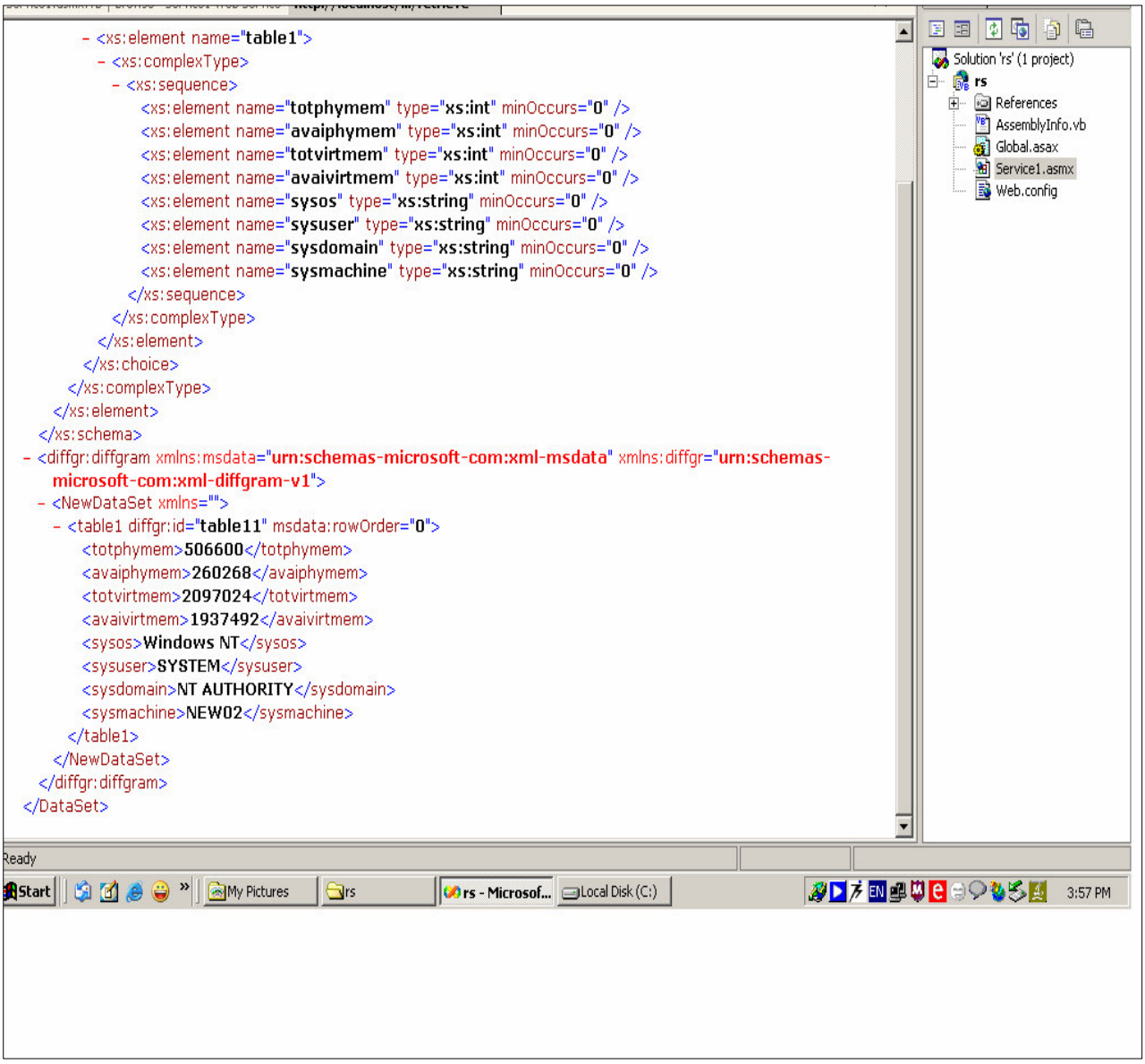
6.3.2 Experimental Setup



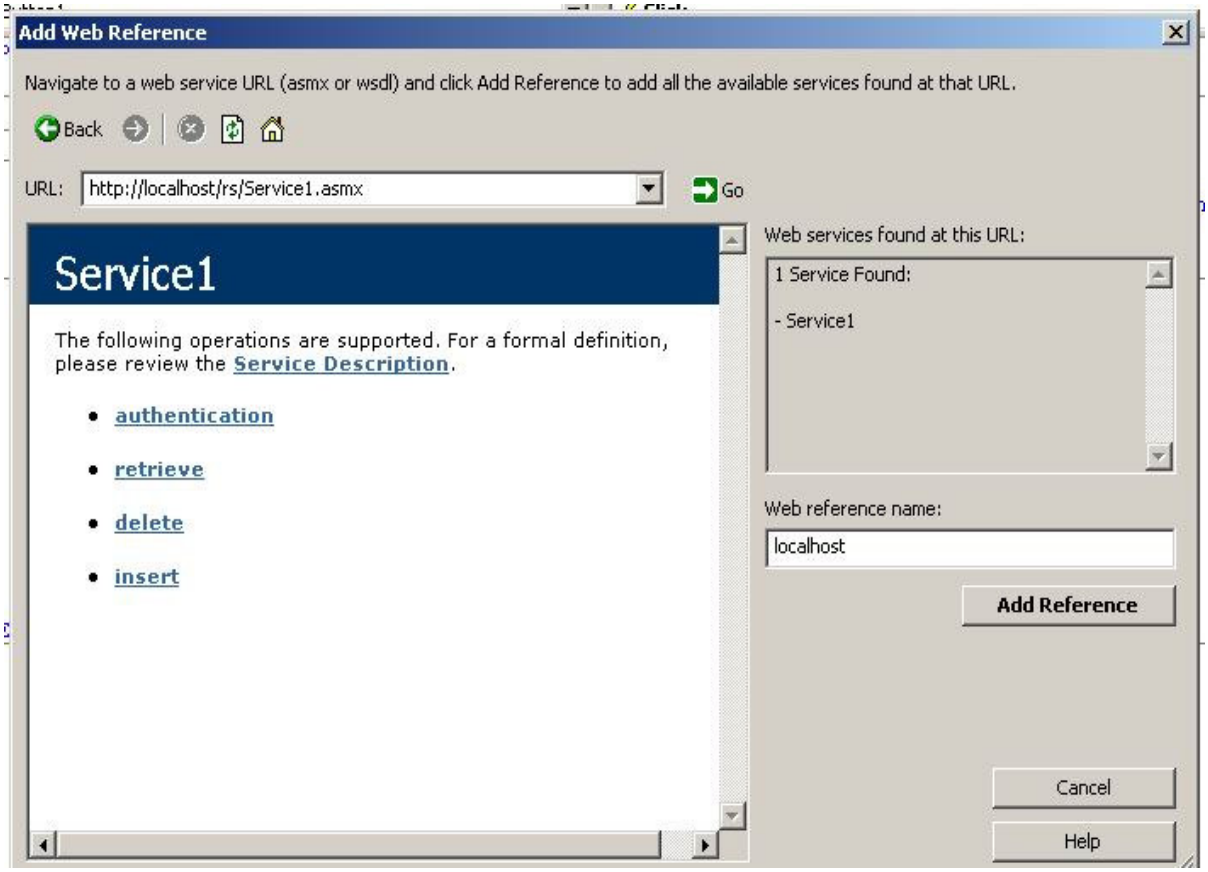
Screenshot6.8: Authentication form of Resource Consumer



Screenshot6.9: Homepage of TU Grid Resource Consumer System



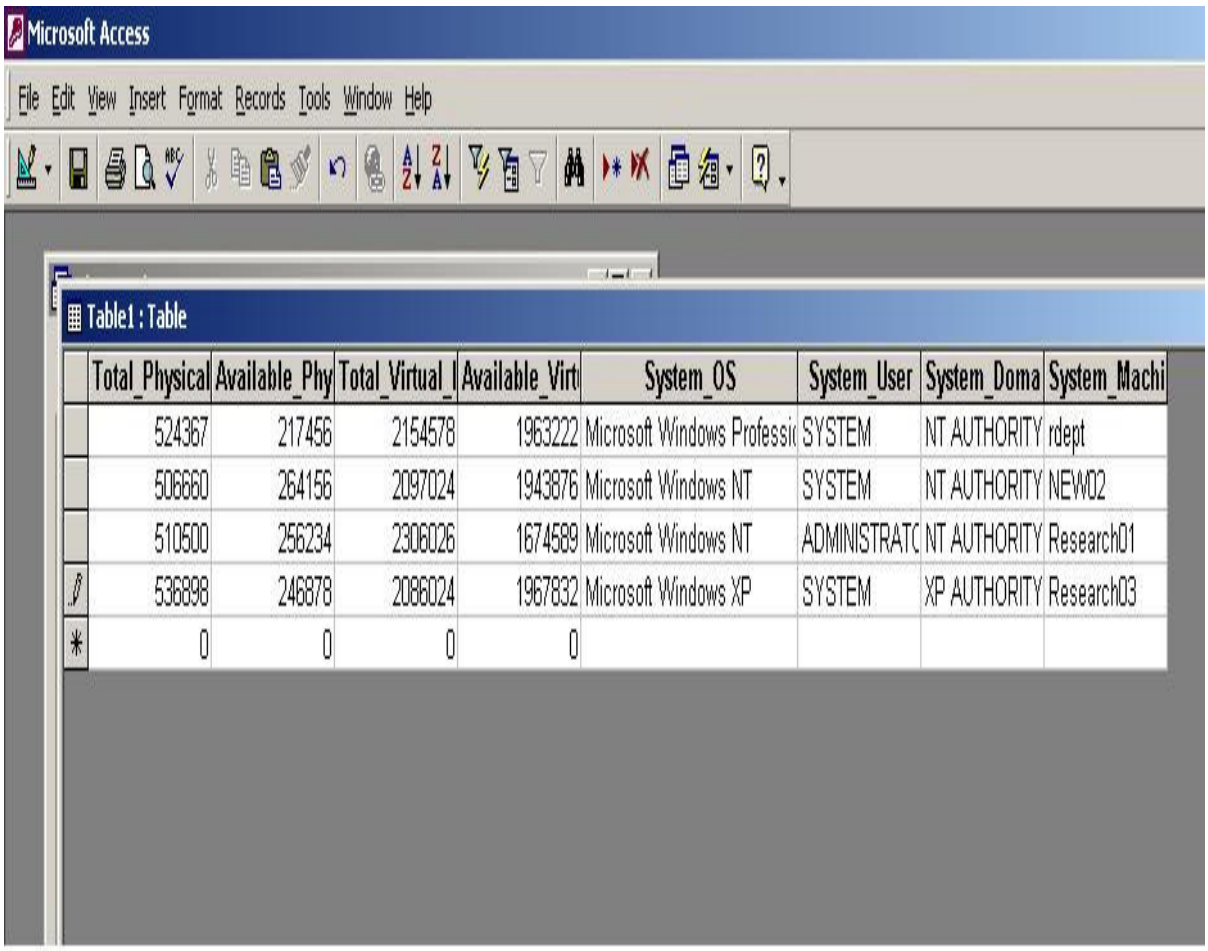
Screenshot6.10: WSDL for web method: retrieve ()



Screenshot 6.11: Consuming Web Service in Web Application

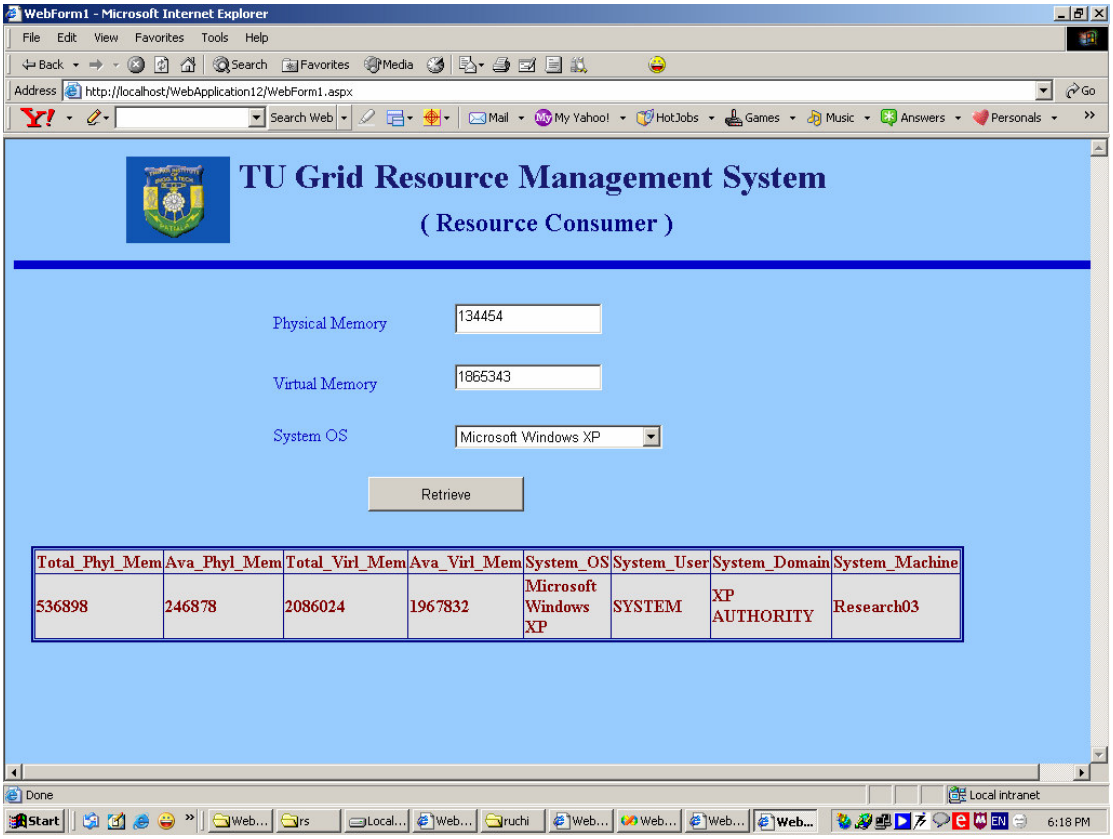
.

6.4 Experimental Results



Total_Physical	Available_Phy	Total_Virtual_I	Available_Virt	System_OS	System_User	System_Doma	System_Machi
524367	217456	2154578	1963222	Microsoft Windows Professi	SYSTEM	NT AUTHORITY	rdept
506660	264156	2097024	1943876	Microsoft Windows NT	SYSTEM	NT AUTHORITY	NEW02
510500	256234	2306026	1674589	Microsoft Windows NT	ADMINISTRATC	NT AUTHORITY	Research01
536898	246878	2086024	1967832	Microsoft Windows XP	SYSTEM	XP AUTHORITY	Research03
0	0	0	0				

Screenshot 6.12: Status Database



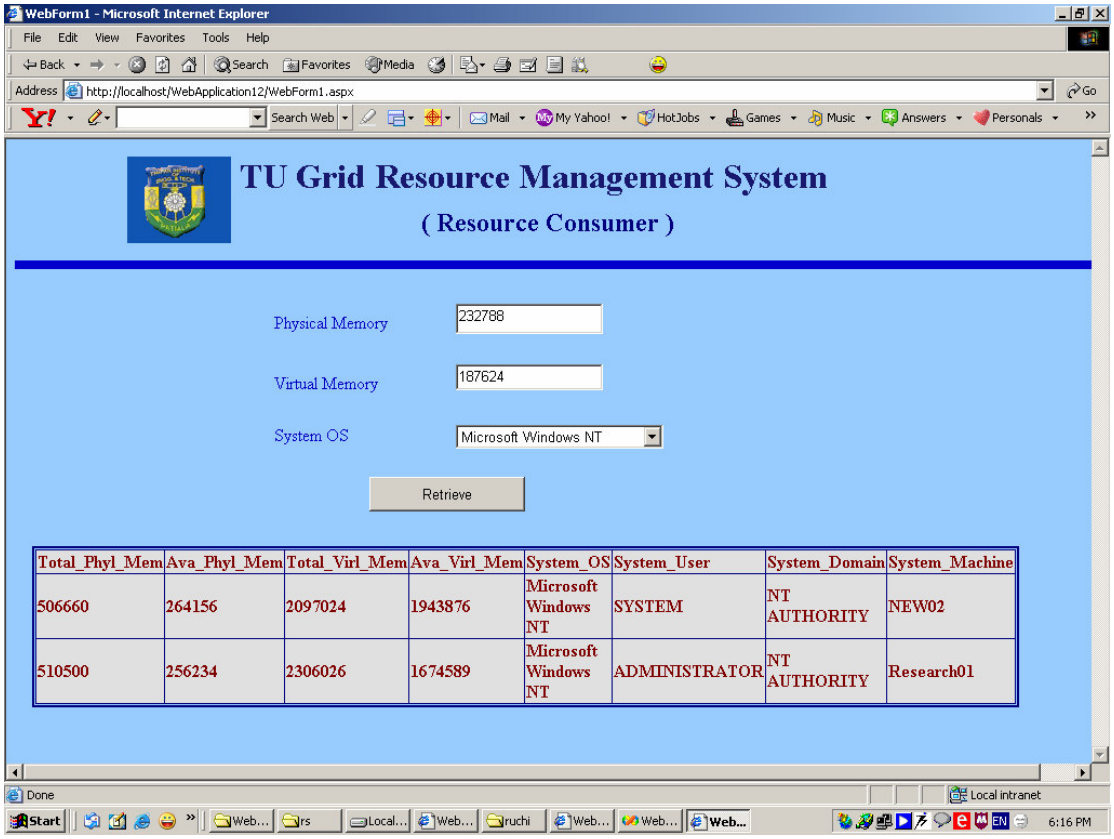
Screenshot 6.13: Experimental result 1

Physical Memory required is 134454 KB

Virtual Memory required is 1865343 KB

Operating System required is Microsoft Windows XP

This image shows list of resources displayed as per constraints specified by the consumer on physical memory, virtual memory and operating system. Operating system is selected from the drop down list as list of available Operating system is fetched from database avoiding syntax mistake in keyword search.



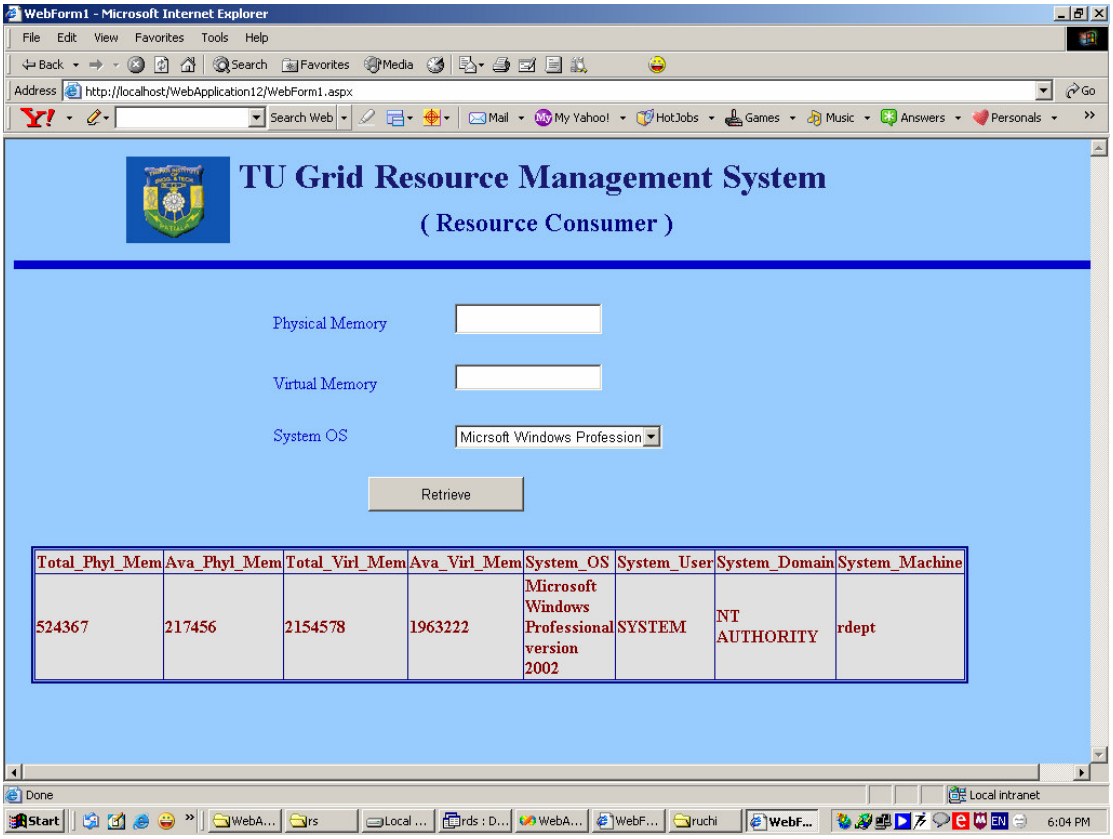
Screenshot 6.14: Experimental Result 2

Physical Memory required is 232788 KB

Virtual Memory required is 187624 KB

Operating System required is Microsoft Windows NT

This image also shows list of resources displayed as per constraints specified by the consumer on physical memory, virtual memory and operating system.



Screenshot 6.15: Experimental Result 3

In this image no constraint is specified for Physical memory and Virtual memory. Minimum value in status database is taken as default value i.e. all the system having Microsoft Windows Professional as Operating system are being displayed. Operating system was selected from drop down list.

6.5 Comparing Alchemi and TU Grid Resource Management System

- The .NET Framework offers two mechanisms for execution across application domains – remoting and web services (application domains are the unit of isolation for a .NET application and can reside on different network hosts).
  .NET remoting allows a .NET object to be "remoted" and expose its functionality across application domains. remoting is used for communication between the four Alchemi (Appendix A) distributed Grid components
  While In this thesis work, Web services concept being used for various advantages listed in previous chapters.

- In Alchemi an Executor, non –dedicated, is that the user manages the resource on a volunteer basis via a screen saver. For non-dedicated execution, there is one-way communication between the Executor and the Manager. In this case, the resource that the Executor resides on is managed on a volunteer basis since it requests threads to execute from the Manager.
  While in this thesis work, client is non-dedicated and allowing usage of its resources on volunteer basis, once resource information is fetched by window service, it is kept at central server.

Conclusion and Future Scope

Resource discovery is one of the most important aspects of ongoing Computational Grid research. Success of Computational Grid depends on locating appropriate resources for a specific task. This report identified several resource discovery algorithms that can be implemented in Computational Grid.

Many of the problems associated with the implementation of Grids could be resolved if the Web is used as the underlying ("backbone") infrastructure for the development of Grids, thus resulting in the formation of a Grid-enabled or a partially Gridified-Web. In this thesis, we propose that Grids should be developed using the underlying Web infrastructure and Grid services could be integrated with Web services. Also this approach has been realized by implementing resource discovery as web service using .NET framework.

Following Issues have been dealt with in this thesis work:

- Capturing a Grid Resource discovery application as a Web service.
- Using Microsoft's .Net Framework to address Windows Based Grid Computing.

The experimental results demonstrate that on the design principles proposed in this thesis work, web service can be effectively used for Grid Resource Discovery.

Future Scope

Grid computing has a lot of vested interest by large companies that can use it to connect geographically diverse offices and unify the supply chain. Web Services are the way to do this. In industrial sector, IBM is the leader in this field and expending myriads of funds. From the above details we can conclude that the future of Grid services, based on the array of current technologies, is very healthy.

The Web Services based Resource discovery for Grid environments described in this thesis can be further integrated with other resource management steps (like resource allocation, resource provisioning) to develop a Web Services based Grid middleware.

References

[1] Heather Kreger (May, 2001), *Web Services Conceptual Architecture (WSCA 1.0),* IBM Software Group, http://www-106.ibm.com/developerworks/library

[2] "*CORBA: Common Object Request Broker Architecture*", http://www.corba.org

[3] Entropia, http://www.entropia.com

[4] Gnutella: http://gnutella.wego.com/

[5] Napster, http://www.napster.com/

[6] Parabon, http://www.parabon.com

[7] I. Foster, C. Kesselman, S. Tuecke, "*The Anatomy of the Grid: Enabling Scalable Virtual Organizations*", Intl. J. Supercomputing Applications, 2001

[8] I. Foster and C. Kesselman, *The Grid: Blueprint for a Future Computing Infrastructure.* Morgan Kaufmann Publishers, USA, 1999.

[9] I. Foster, C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, Calif. (1999).

[10] I. Foster, C. Kesselman, S. Tuecke, *Int. J. High Perform. Comput. Appl.* 15(3), 200 (2001). Also available at http://www.globus.org/research/papers/anatomy.pdf.

[11] I. Foster. What is the Grid? A Three Point Checklist. GRIDToday, July 20, 2002.

[12] Foster, I., Kesselman, K., Nick, J., Tuecke, S., "The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration", in *Grid Computing: Making the Global Infrastructure a Reality*, Fox, G. Ed. Wiley, 2003.

[13] Johnston, "Implementing Production Grids", in *Grid Computing: Making the Global Infrastructure a Reality*, Fox, G. Ed. Wiley, 2003.

[14] I. Foster, C. Kesselman, J. Nick, S. Tuecke: " The Anatomy of the Grid: Enabling Scalable Virtual Organization.Accessed from:

[15] http://www.mnlab.cs.depaul.edu/seminar/fall2002/GridAnatomy.pdf.

[16] A. van Dorp, A. de Mes, E. Rongen, T. van Velzen, R. van Wel., Enterprise Application Integration with Web Services (Technical Report), January 2002.

[17] *Creation and Usage with WebSphere Studio Application Developer* (IBM Redbook) UDDI Community, http://www.uddi.org/community

[18] *A Primer to HTTPR*, http://www.ibm.com/developerworks/webservices/library/ws-http/

[19] S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, R. Neyama: *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI.* Sams Publishing, January 2002.

[20] *Apache SOAP,* http://xml.apache.org

[21] *apache AXIS:* http://xml.apache.org/axis/index.html

[22] *Universal Description Discovery and Integration (UDDI), Technical White Paper*, (September 6, 2000), http://www.uddi.org

[23] S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, R. Neyama: *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI.* Sams Publishing, January 2002.

[24]*Web Services Description Language (WSDL),*
http://www-106.ibm.com/developerworks/library/w-wsdl.html

[25] *WSDL4J Open Source*, http://oss.software.ibm.com/developerworks/projects/wsdl4j

[26] *Web Services Development Toolkit* (WSTK): http://www-106.ibm.com/developerworks

[27]*Web Services Development Environment* (WSDE):
http://www-106.ibm.com/developerworks

[28] *Web Services Flow Language (WSFL),*
http://www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[29] eather Kreger (May, 2001), *Web Services Conceptual Architecture (WSCA 1.0),* IBM Software Group, http://www-106.ibm.com/developerworks/library

[30] Napster, http://www.napster.com/

[31] Gnutella: http://gnutella.wego.com/

[32] I.Clarke, O. Sandberg, B.Wiley, and T. W. Hong, "*Freenet: A Distributed Anonymous Information Storage and Retrieval System*", ICSI Workshop on Design Issues in Anonymity and Unobservability, 1999.

[33] SETI@home, http://setiathome.ssl.Berkeley.edu

[34] Parabon, http://www.parabon.com

[35] Entropia, http://www.entropia.com

[36] L. Gong, Project JXTA: A Technology Overview, Technical Report, Sun Microsystems Inc., April 2001, http://www.jxta.org/project/www/docs/TechOverview.pdf

[37] Mazumdar, S; *Mapping of Common Management Information Service (CMIS) to CORBA Object Services*; Bell Labs.; September 1996;

[38] *Java Remote Method Innovation*,
http://java.sun.com/products/jdk/rmi/http://java.sun.com/products/jdk/rmi

[39] M. B. Juric, I. Rozman, M. Hericko, T. Domajnko: *CORBA, RMI and RMI-IIOP Performance Analysis and Optimization*. University of Maribor, Slovenia. 2001

[40] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran "A Taxonomy and Survey of Grid Resource Management Systems", School of Computer Science and Software Engineering,Monash University,Melbourne, Australia.

http://www.buyya.com/papers/Gridtaxonomy-report.pdf

[41] Manfred Hauswirth, Roman Schmidt ,"An overlay network for Resource Discovery in Grids," http://dip.semanticweb.org/documents/An- overlay-network-for-resource-discovery- in-Grids.pdf

[42] Tierney, B. et al., "A Grid Monitoring Architecture," Global Grid Forum, Lemont,Illinois,U.S.A.,January2002.
http://www.Gridforum.org/documents/GFD.7.pdf

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

[44] http://www.ggf.org/ggf_Grid_understand.htm

[45] Adriana Iamnitchi and Ian Foster, "A Peer-to-Peer Approach to Resource Location in Grid Environments ," University of Chicago.

[46] 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.

[47] 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.

[48] 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.

[49] I. Foster and C. Kesselman, "The Globus Project: a Status Report", In Proc. of Seventh Heterogeneous Computing Workshop (HCW 98), IEEE Computer Society Press, March, 1998.

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

[51] 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.

[52] Naseer, A., and Stergioulas, L.K. Resource Discovery in Grids and Other Distributed Environments: States of the Art. To appear in Multiagent and Grid Systems - An International Journal, IOS Press, ISSN: 1574-1702, 2006.

[53] Heather Kreger (May, 2001), *Web Services Conceptual Architecture (WSCA 1.0),* IBM Software Group, http://www-106.ibm.com/developerworks/library

[54] A. van Dorp, A. de Mes, E. Rongen, T. van Velzen, R. van Wel., Enterprise

Application Integration with Web Services (Technical Report), January 2002.

[55] Fox, G., Balsoy, O., Pallickara, S., Uyar, A., Gannon, D. and Slominski, *A. Community Grids*. Community Grid Computing Laboratory, Indiana University, 2002

[56] G. Fox, M. Pierce, S. Ko, etc., *Grid Services for Earthquake Science*, Indiana University, Florida State University, etc. 2001

[57] G. Fox, S. Pallickara, *The Narada Event Brokering System: Overview and Extensions*. Community Grid Labs, Indiana University. 2001

[58] I. Foster. What is the Grid? A Three Point Checklist. GRIDToday, July 20, 2002

[59] Franccois Grey, Matti Heikkurinen, Rosy Mondardini, "*Grid Cafe: A Palace for Everybody to Learn About Grid*"

http://Gridcafe.web.cern.ch/Gridcafe/Gridwork/architecure.html

[60] http://www.microsoft.com/com/default.mspx

[61] A short introduction to Web Services

http://gdp.globus.org/gt4-tutorial/multiplehtml/ch01s02.html

Appendix A
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.

List of Publications

1. Ruchi Garg, Seema Bawa, "Grid Resource Discovery using Web Services", **IEEE Student Chapter Conference**: TROIKA'07, 21 February, 2007, at Delhi College of Engineering, Delhi.

2. Ruchi Garg, Seema Bawa, "Web Services based Grid Resource Discovery" in Proceedings of National Seminar on Business Transformation Through Technological Integration ISTE Sponsored, PIMT Mandi Gobindgarh, 30 March,2007.