2005

Status based resource discovery in computational grids.

Mohammad Aktaruzzaman
University of Windsor

NOTE TO USERS

Status Based Resource Discovery in Computational Grids

by

Mohammad Aktaruzzaman

A Thesis

Submitted to the Faculty of Graduate Studies and Research

through the School of Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University *of* Windsor

Windsor, Ontario, Canada

2005

Your file Votre référence
ISBN: 0-494-04972-3
Our file Notre référence
ISBN: 0-494-04972-3

Abstract

Resource discovery is an important aspect of Computational Grids. Locating resources in a Grid environment is difficult because of the geographic dispersion and dynamic nature of its resources. Issues such as large numbers of users, heterogeneous resources and dynamic status of resources over time in a large distributed network make resource discovery more difficult than the case of traditional networks. In the case of Computational Grids, additional issues such as different operating systems, different administrative domains and lack of portability between platform dependent applications make resource discovery even more difficult. Further to all these difficult issues, knowledge of the current status of the resources adds an extra challenge to the problem of finding resources in the Grids. An ideal Computational Grid environment should contain a resource discovery infrastructure that includes heterogeneous resource monitoring capabilities. These capabilities will save time and the risk of selecting inappropriate resources.

In this thesis work, we propose a resource discovery infrastructure in the form of an automated status monitoring model. The model consists of two fundamental aspects, a portable data model and a set of executable monitoring components. Our approach adheres to principles of software design, is well structured and platform independent. The portable data model, which conveys the status of the resources, must be understandable by any application software, agent or scheduler on any platform. In turn, the monitors must be able to acquire necessary status information from various, diverse systems and maintain the data model. We developed appropriate interfaces that provide straightforward connectivity between our infrastructure and other Grid middleware components being developed elsewhere.

Key words: Computational Grids, resource discovery, status discovery, heterogeneous environment, small world, distributed computing

Dedication

To my lovely wife Lisa, my lovely and magnificent son Aryan, and my mom and dad.

Acknowledgements

I would like to gratefully acknowledge the wonderful and enthusiastic supervision of Dr.Robert Kent. I would like to thank Dr.William Baylis, Dr. Boubakeur Boufama, and Dr. Dan Wu for their valuable comments and support. I am grateful to all of my friends for their endless support and co-operation.

Table of Contents

List of Tables

List of Illustrations

1 Introduction

A computational Grid is a new class of infrastructure, which can provide scalable, secure, high performance solutions for discovering and accessing remote resources across the globe [Fost02a] . Generally Computational Grid is a collection of different resources across the world and the foundation blocks of such systems are mainly personal computers, desktops, and the need of sharing resources of the resource providers. The need of building a Computational Grid is to achieve something which is not possible to achieve for a low end machine. Since Computational Grid is a large distributed network, it is not easy to find desired resource, because of the heterogeneous nature of the resources, large number of users, different administrative domains, and different operating systems.

Current status of the resources also plays an important role in Computational Grid in order to avoid choosing an occupied resource. It would be nice if the service requestor gets a list of all service providers along with the current status of the resources for a specific task. Here status of the resources refers to the current CPU usage, total memory, available memory, type of operating systems, and a list of available services. The conventional resource discovery solutions do not deal with status monitoring of the resources properly. A conventional resource discovery solution only discovers the resource, not the status of the resource. But if the service requestor knows the status of the desired resources before hand, it would be easier to complete a specific task.

An ideal Computational Grid environment should contain a resource discovery infrastructure along with heterogeneous resource monitoring capabilities. These capabilities can reduce the risk of choosing a wrong resource and also can reduce the

completion time of a given task. The capabilities also should be user friendly in nature, because a large portion of the Grid community is end-users and most of them do not have expert technical knowledge to operate different complicated systems.

1.1 The Problem

Grid Computing is a new idea in distributed computing. The success of grid computing depends on the collaboration of different resources across the globe. Not only the collaboration among resources brings the success to Grid Computing, appropriate resource discovery technique plays the very important role in connecting different resource providers. The general members of a Computational Grid are low end computers, personal computers which are owned by regular users. We have to remember one thing that member computers are not fully dedicated to the grid community. These resources are meant for regular use and they are available for grid community during their idle time.

As an owner of such system, nobody wants any interruption in the middle of their work in the name of searching for an appropriate resource. Because an interruption in middle of work can slow down the processing time of the desired task, it may also freeze the computer. In the resource discovery techniques, which are available for Grid Computing, normally resource requestors search for the resource directly in the resource providers. But recent internet security concern makes this technique complicated. Therefore, using a firewall has become a very popular technique of protecting the computer from the outside world. But firewall is very conservative about most of the incoming traffic.

1.2 The Challenge

The main challenge of existing grid technology we are facing now is lack of availability of a well engineered software infrastructure. Grid technology is for the people not only for dedicated computer expert. As for example, if we consider the example of Globus [Fost97], which is one of the most popular but least user friendly infrastructure available in the Grid arena. In order to use this system, a user has to be very expert in Linux, and this is available only for Linux platform.

The second big challenge we are currently facing is portability among different platforms, in other words the homogeneous - heterogeneous issue. Because we have windows users, we have Linux users, we have UNIX users, and we have Mac users. Although they have different names, and different file systems, the users who use these platforms their needs are pretty much same; their work has to be done. This is something like we speak different languages, but we eat food to keep us healthy, and what kind of foods we eat that is a different issue. In the same way, if a user needs to do a job, that user does not care which system hosts the resource, windows, Linux, UNIX and in order to complete the job if different platforms need to communicate, they should be able communicate without any communication overhead.

In this thesis work a portable, platform independent and well engineered system is presented.

1.3 Contribution

The main contributions of this research work are given below:

a) Well-engineered, heterogeneous resource monitoring architecture.

b) In this system, the resource provider will provide the information about resources in timely manner. So the service provider will not be getting interrupted when he/she is busy with his/her work.

c) CPU/Memory friendly solution: the application consumes less CPU and less memory.

d) The maintenance and production of the system is reasonable.

e) This architecture may be integrated with other existing Grid architecture.

1.4 Chapter Organization

The thesis is divided into five main chapters, as follows. Chapter 2 presents some related works. Chapter 3 presents the various design decisions made, implementation strategies, implementation details and their justification. Chapter 4 presents the experimentation and results. Chapter 5 concludes the thesis with a summary and the discussion of future work.

2 Previous / Related Work

This chapter explores selected research efforts related to resource discovery in Grid computing. Several resource discovery architectures, several protocols have been presented in this chapter.

2.1 General Discussions

The following terms, components, and theory have been reviewed to conduct this thesis work:

2.1.1 Computational Grid

As stated in [Fost02a], and [Fost98] the computational grid is "a new class of infrastructure, which provides scalable, secure, high-performance mechanisms for discovering and negotiating access to remote resources, the Grid promises to make it possible for scientific collaborations to share resources on an unprecedented scale, and for geographically distributed groups to work together in ways that were previously impossible." According to [Fost98], in 1965, Fernando Corbato and other designers of the Multics operating system from MIT had foreseen a computer system, which is similar to a power grid or a water grid. In 1968, J. C. R. Licklider and Robert W. Taylor predicted a Grid-like infrastructure in their paper "The Computer as a Communications Device".

According to [Iamn01], the resources are computers, cluster of computers, online instruments, storage space, data, application and a resource discovery mechanism returns the identity (may be location address(es)) of matching resources for a given description of desired resources.

A large number of users, heterogeneous resources, dynamic status of resources (over time) in a large distributed network make resource discovery more difficult than that of traditional network. In a Computational Grid, Different operating systems, different administrative domains, lack of portability between platform dependent applications make resource discovery more difficult. So, appropriate resource discovery mechanism is an important aspect of Grid Computing. Success of a Computational Grid mainly depends on locating appropriate resources for a specific task.

As stated in [Fost02c] , a Grid is a system, which has following three characteristics:

- ***"coordinates resources that are not subject to centralized control ...***

A Grid integrates and coordinates resources and users that live within different control domains—for example, the user's desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system."

- ***"using standard, open, general-purpose protocols and interfaces***

A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access.."

- ***"to deliver nontrivial qualities of service***

A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet

complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts."

2.1.2 Virtual Organization (VO)

As stated in [Fost01], Virtual organization refers to "secure, flexible, coordinated resource sharing among dynamic collections of individuals, institutions, and resources."

2.1.3 Components of Resource Discovery

[Iamn02c], and [Iamn02a] identify four different architectural components for a general resource discovery solution, "Membership protocol", "Overlay construction", "Pre-processing", "Request processing". [Iamn02a] also identifies four environment parameter factors, which dominate the performance and design strategies for a resource discovery solution. These four factors are "Resource information distribution and density", "Resource information dynamism", "Request popularity distribution", "Peer participation".

2.1.3.1 Architectural Components

According to [Iamn02c], and [Iamn02b], the following architectural components are identified:

2.1.3.1.1 Membership Protocol

It refers to how a new node becomes a member of an unstructured network and how a member learns about other members in such network. It is responsible for collecting and updating information about currently active members.

2.1.3.1.2 Overlay Construction Function

Overlay construction function refers to select the active members from the local membership list. This selected member list depends on bandwidth availability, network load, administrative policies, and topology specifications. It is responsible for searching the best match for a given request.

2.1.3.1.3 Pre-processing

Pre-processing refers to off-line preparations for achieving better search performance. An example of pre-processing is an advertisement for a local resource to other networks. It is responsible for efficient searches.

2.1.3.1.4 Request Processing

It is responsible for searching resources. It has two components:

Local Processing:

It facilitates looking up local resource information for a requested resource.

Remote Processing:

It refers to send the request to other networks through different mechanism for appropriate resources.

2.1.3.2 Architectural Modeling

According to [Iamn02c][Iamn02a], [Iamn02a], the following four environment parameter factors are identified:

2.1.3.2.1 Resource Information Distribution and Density

It refers to different load of information sharing on different nodes. As for example, some nodes share a large number of resource information, on the other hand, a home computer shares just a few. Some resources are very common and some are very unique and rare.

2.1.3.2.2 Resource Information Dynamism

It refers to different attributes of resources, which are dynamic and static in nature. For example, CPU load, memory availability are two highly variant attributes. On the other hand, operating systems, type of CPU are two nearly static attributes.

2.1.3.2.3 Requests Distribution

It refers to possible closer uniform distribution of popular requests. L. Breslau and others showed in their paper "Web caching and zipf-like distributions: Evidence and implications" that HTTP requests follow zipf distributions.

2.1.3.2.4 Peer participation

The peer participation depends on different type of networks. For example, peer participation in p2p networks significantly varies as in Computational Grid.

2.1.4 Models for grid resource management

[Buyy00b] discusses three different management models for grid resource management mainly hierarchical model, abstract owner model and (computational) market model. Hierarchical model contains different active and passive components for grid computing. As for example "Domain Control Agents" is an active component, which can provide state information through publishing in an information service or through direct query. Abstract owner model refers to an order and delivery and result gathering approach. The (computational) market model refers to mixture of both hierarchical and abstract owner

model and it refers to a computational economy development in grid resource management. The authors believed that one or more of the discussed models could be mapped with existing or future grid systems.

[Krau00] identifies some key resource management approaches to design a comprehensive resource management system. In this paper, the authors presented resource management system as the core component of a network computing system (NCS). The authors discussed different aspects of resource management such as mainly quality of service (QoS) issue, different heterogeneity issues, different scheduling approaches, resource discovery technique, different resource distribution approaches. The authors suggest that a resource management system can maintain a replicated network directory, which may contain resource information, and then the resource discovery function queries the "resource dissemination function" for a particular resource.

2.2 Protocols

The following protocols are identified in [Alle01], [Alle99], [Chan02], [Czaj01], [Czaj98], [Fost97], [Lars01], [Rosz98], [Fost01], [Fost02b], [Verb02] :

2.2.1 Grid Information Protocol (GRIP)

According to [Czaj01], Grid Information Protocol (GRIP) is used to retrieve information of entities in virtual organization [Alle01],[Fost01] .According to [Czaj01], this protocol is one of the main building blocks of VO. [Alle01] proposes an adaptive resource selection mechanism to facilitate autonomous application migration for better resources due to degradation of resources during the execution, and they use Grid Registration Protocol (GRRP) of Globus Monitoring and Discovery Service (MDS) to discover the resource.

2.2.2 Grid Registration Protocol (GRRP)

According to [Czaj01], Grid Registration Protocol (GRRP) is responsible for notifying the aggregate directory services which refer to VO-specific resources, which are found by GRIP protocol. Both GRIP and GGRP are used by the Globus metacomputing toolkit [Fost97]. [Alle01] uses GRIP protocol of Globus Monitoring and Discovery Service (MDS) to discover the resource in the proposed adaptive resource selection mechanism. [Alle01] claims that this service queries appropriate aggregate (e.g. GRRP) to discover a "potentially interesting" resource and it uses GRIP to locate that resource during the execution of the application.

2.2.3 Lightweight Directory Access Protocol (LDAP)

LDAP protocol is used for retrieving and updating information in a X.500 (to store information in a directory based on certain regulations) model based directory. According to [Fost01], a LDAP server speaks LDAP protocol, and it also makes a response to the query. [Czaj98] shows how a Lightweight Directory Access Protocol (LDAP) enabled Globus Monitoring and Discovery Service (MDS) can be used as an information service, that is responsible for providing current availability and capability of resources. "Project Isaac", also uses a Lightweight Directory Access Protocol (LDAP), that was introduced by [Rosz98].

2.2.4 Common Indexing Protocol (CIP)

According to [Alle99], the Common Indexing Protocol (CIP) is used to pass indexing information from server to server to facilitate query routing or redirecting to a client. Project Isaac also uses common Indexing Protocol (CIP) [Rosz98] .

2.2.5 Z39.50

According to [Lars01], Z39.59 is an information retrieval protocol, which was developed by the American National Information Standards Organization (NISO). [Lars01] presents a development technique of developing a cross-domain resource discovery database using the Z39.50 protocol and they use Z39.50 "Explain Database" to select the databases and to search the "Explain Database" to extract the server information using a probabilistic algorithm to retrieve and rank the collection of information to the user for selection.

2.2.6 Network-Efficient Vast Resource Lookup At Edge (NEVRLATE)

[Chan02] proposes a protocol called NEVRLATE (Network-Efficient Vast Resource Lookup At Edge) that facilitates a scalable resource discovery service, and it is used for vast resource discovery. NEVRLATE organizes the directory service mainly in two-dimensional Grids: registration to the resources is stored in one horizontal dimension and lookup is performed in vertical dimension. The main specialty of this protocol is to organize n servers into a flexible structure takes an $O(\sqrt{n})$ message complexity mainly for registration and discovery of the resource. They claimed that NEVRLATE service could provide a scalable worldwide "semantic" and "extended web" infrastructure.

2.2.7 JXTA

As stated in jxta.org [JXTA], "JXTA protocols are a set of protocols, which are designed for ad hoc, pervasive, and multi-hop peer-to-peer (P2P) network computing. Using the JXTA protocols, peers can cooperate to form self-organized and self-configured peer groups independent of their positions in the network (edges, firewalls, network address translators, public vs. private address spaces), and without the need of a centralized

management infrastructure." [Verb02] presents a framework for large-scale computations for "coarse-grained" parallelization. The components of this proposed framework are based on JXTA protocol, which can facilitate a dynamic and decentralized organization of computation resources. In this framework, JXTA protocol facilitates dynamic aspect of grid through peer discovery where nodes are added or removed during the job execution.

2.2.8 Simple Object Access Protocol (SOAP)

According to [Fost02b], SOAP is an enveloping mechanism for carrying XML payloads. This protocol provides a message passing mechanism to exchange information between service provider and requestor. SOAP payload can be carried HTTP, FTP, JMS (Java Messages Service), etc.

2.3 Approaches

Following resource discovery mechanisms are reviewed in [Hosc02b], [Harc99], [Huan02], [Iamn01], [Iamn02b], [Iamn02a], [Jun00], [Li02], [Ludw02], [Mahe00], [Rana01], [Wols99]:

2.3.1 Peer-to-Peer Approach

[Iamn01], [Iamn02b], and [Iamn02a] discuss peer-to-peer resource discovery in detail. [Iamn01] proposes peer-to-peer resource discovery architecture for a large collection of resources. [Iamn01] claims that this decentralized resource discovery architecture could lessen huge administrative burden as well as it can also provide very effective search-performance result. They analyzed this resource discovery mechanism on up to 5000 peers based on the assumption that every peer provides at least one resource.

[Iamn02a] discusses different resource discovery problems in a large distributed resource-sharing environment specially in a grid environment. In this document, author identified four different architectural components called "Membership protocol", "Overlay construction", "Pre-processing", and "Request processing". Author also identified four environment parameter factors, which dominate the performance and design strategies for a resource discovery solution. These four factors are "Resource information distribution and density", "Resource information dynamism", "Request popularity distribution", "Peer participation".

[Iamn02b] gives a brief description of different resource discovery approaches in peer-to-peer networking. The authors claimed that using four axes framework [Iamn02a], it is possible to design any resource discovery architecture in a grid.

[Hosc02b] proposes a general purpose query support enabled "Unified Peer-to-Peer Database Framework (UPDF)" for large distributed systems. UPDF can be identified as a peer-to-peer database framework for a general purpose query support which is unified because it supports arbitrary query languages, random node topologies, different data types, different query response modes, different neighbour selection policies for expressing specific applications.

2.3.2 De-Centralized Approach

[Iamn01] and [Rana01] discuss this approach. [Rana01] describes a de-centralized resource management and discovery architecture based on interacting software agents where agents can represent as a service, an application, a resource or a matchmaking service. The authors of [Rana01] shows that this proposed approach could provide dynamic registration of resources and user task. According to this paper, this approach is

a matchmaking approach, which can facilitate dynamic resource management and resource discovery in a grid environment. It also provides the resource availability, resource capability.

2.3.3 Agent-Based Approach

[Jun00] presents a distributed resource discovery method for a large wide area distributed system. This paper presents different resource discovery algorithms mainly flooding algorithm, swamping algorithm, random pointer jump algorithm, and name dropper algorithm. After analyzing all these algorithms the authors proposed a new agent-based resource discovery algorithm called "Distributed Awareness Algorithm" and they also proposed a framework for dynamic assembly of agents based on this algorithm. Distributed awareness refers to a learning mechanism in which a node gets awareness about other nodes in a network. In this algorithm, each node has an awareness table and each node exchanges the information of this awareness table with other nodes. A typical awareness table entry contains location of the node (IP address), when last heard from that node, when last time the awareness information was sent to the node.

They also claimed that this agent based resource discovery system can provide better discovery services using its agents' autonomous behaviour and some other existing technology including "Bond agents", and some existing JPython written resource monitoring software.

2.3.4 Ontology Description- Based Approach

Ontology refers to a description of a service (resource); [Ludw02] proposes a semantic service discovery framework in a grid environment. They propose a service matchmaking mechanism based on ontology knowledge and they claimed that this matchmaking

framework can provide a better service discovery and also can provide close matches. The main idea behind this approach is the advertisement of the resource. In this approach, service provider registers its service description into the service registry database. When a Grid application sends a request to service directory, matchmaker returns the matches to the service requester. Requester chooses the best resource based on the specific need.

2.3.5 Routing Transferring Model-Based Approach

[Li02] proposes a resource discovery technique called the Routing-Transferring Model. This model consists of three basic components - resource requester, resource router and resource provider. The provider sends the resource information to a router and router stores that information in a router table. After that, when the requester sends a request to the router, router checks its routing table for an appropriate resource provider and after finding that entry router forwards that request to the service provider or another router. The authors formalized this model and they analyzed the complexity of Shortest Distance Routing Transferring (SD-RT) algorithm based on this formalization. They claimed that resource discovery time depends on topology and they also showed that SD-RT could locate a resource in the shortest time, if the topology and distribution of resources are explicit. They examined their proposed model in Vega Grid project and their experiment shows that higher frequency and more location of resources can reduce the resource discovery time.

2.3.6 Parameter-Based Approach

[Mahe00] examines different approaches for resource discovery in a grid system. A new concept "Grid potential" is proposed in this paper, which encapsulates the processing capabilities of different resources in a large network. The authors also proposed an

algorithm called "Data Dissemination Algorithm". This algorithm follows swamping approach [Harc99] 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 that permits all incoming messages, neighbourhood awareness that allows messages from a certain distance, and distinctive awareness, which discards messages if it finds out that the less Grid potentiality at the local node in remote node, is less than that of the requestor node. The authors also measured the performance of "universal awareness", "neighbourhood awareness", and "distinctive awareness" dissemination schemes. The authors claimed that universal approach is more expensive in terms of message complexity than that of neighbourhood and distinctive approach. The authors also claimed that this new class of dissemination could reduce the communication overhead during the resource discovery.

2.3.7 Quality of Service (QoS)-Based Approach

[Huan02] proposes an algorithm to discover the occasionally available resources in a multimedia environment. In this paper, the authors defined different policies for a QoS based resource discovery service for a given graph theoretic approach. They introduced a generalized version of Discovering Intermittently Available Resources (DIAR) algorithm based on occasionally available resources. They evaluated the performance of QoS policies based on different time-map strategies in a centralized system. Through the experiment they found out randomized placement strategies and increased server storage can facilitate better performance to discover a particular resource.

2.3.8 Request Forwarding Approach

According to [Iamn02b], and [Iamn02a], following four-request forwarding approaches are reviewed:

2.3.8.1 Random Walk Approach

In this approach, to forward the request, the node is chosen randomly.

2.3.8.2 Learning-Based Approach

As discussed in [Iamn02b], and [Iamn02a], a request is forwarded to a node who answered the similar request before. If no similar answer is found, the request is forwarded to a randomly chosen node.

2.3.8.3 Best-Neighbour Approach

The number of received answer is recorded without recording the type of requests. The request is forwarded to that node which answered highest number of requests.

2.3.8.4 Learning-Based + Best-Neighbour Approach

This approach to identical to learning-based approach except when no similar answer is found, request is forwarded to the best neighbour.

In [Iamn02b], and [Iamn02a], the authors analyzed this resource discovery mechanism in an "emulated" grid, which is a large grid network (for this case up to 5000 peers) based on the assumption that every peer provides at least one resource. The authors measured performance evaluation of a simple resource discovery technique based on "request propagation". The authors found that learning-based approach performs better among four request propagation approaches. The authors also claimed that best-

neighbour approach works well in an unbalanced distribution, and although random walk approach performs well in equally distributed resources, but it performs satisfactory in all cases.

2.4 Frameworks/Architectures

2.4.1 Globus

[Fost97] describes a metacomputing infrastructure toolkit called Globus, which was originally developed to integrate geographically distributed resources including supercomputer, cheap desktop, large databases, storages, scientific tools and together they can form distributed virtual supercomputers. In Globus toolkit, basic low level mechanism such as network information, communication, authentication are provided along with high level metacomputing services such as parallel programming tools(MPI) and different schedulers (DUROC). The authors presented this work as an initiative to achieve a large target mainly developing an Adaptive Wide Area Resource Environment (AWARE), which was described as a set of high-level services, an appropriate infrastructure for dynamically changing behaviour of metacomputing environments. Globus uses GRRP and GRIP protocols to discover the resources in VO.

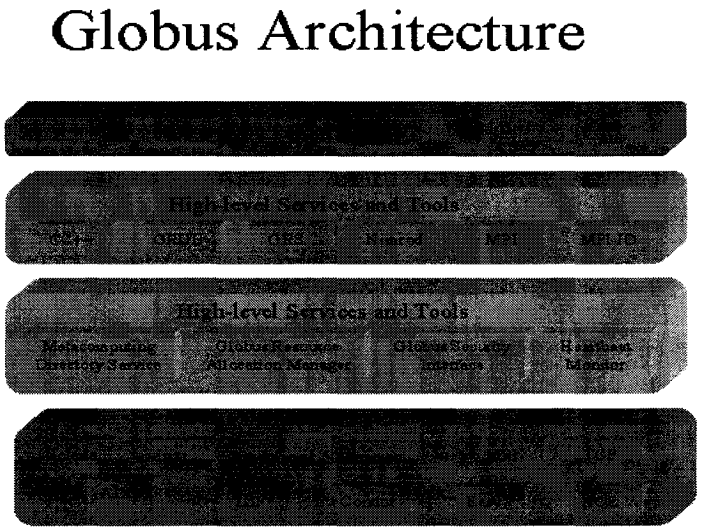

Figure 1: Globus architecture at a glance. [Globus]

2.4.2 Web Service

As stated in [WSA04], "*a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards*"

"*A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided. Although the agent may have changed, the Web service remains the same.*"

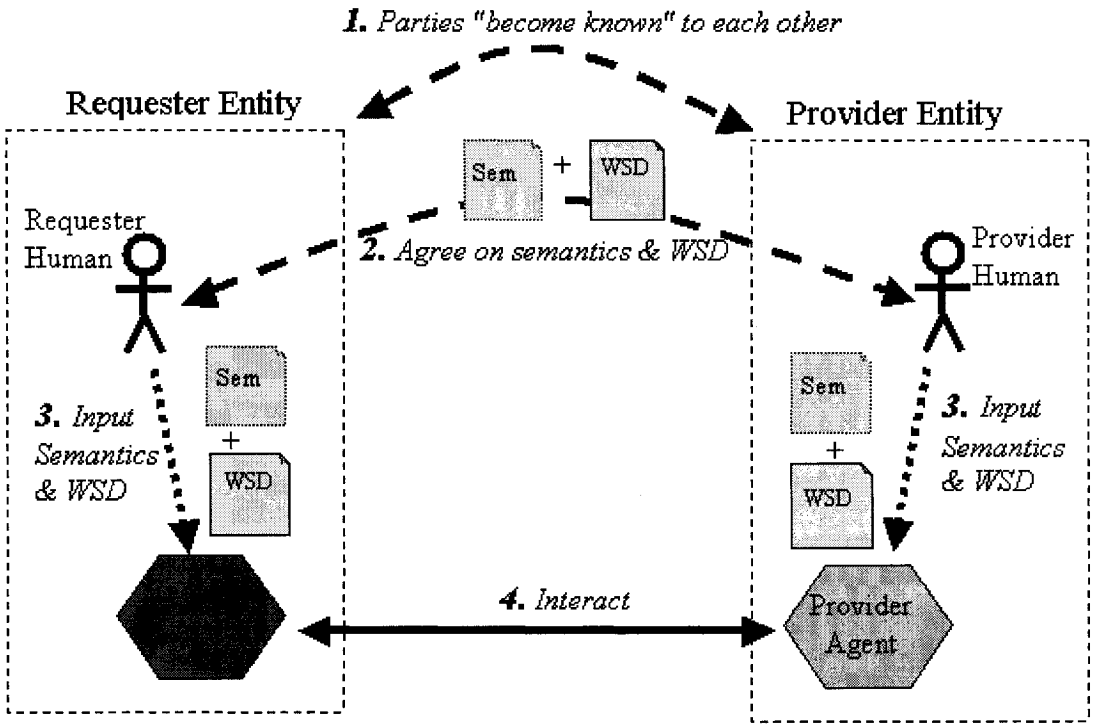

Figure 2: A general scenario of a Web Service's request processing [WSA04]

2.4.3 OGSA

[Hosc02a] proposed a unified and modular service discovery architecture for Grid computing, called Web Service Discovery Architecture (WSDA), which can be used in run time to discover and adapt appropriate remote services. WSDA facilitates an interoperable web service discovery layer by defining industry standard appropriate services, interfaces, and protocol bindings. The communication primitives facilitate service identification, retrieval of service description in a Grid computing environment.

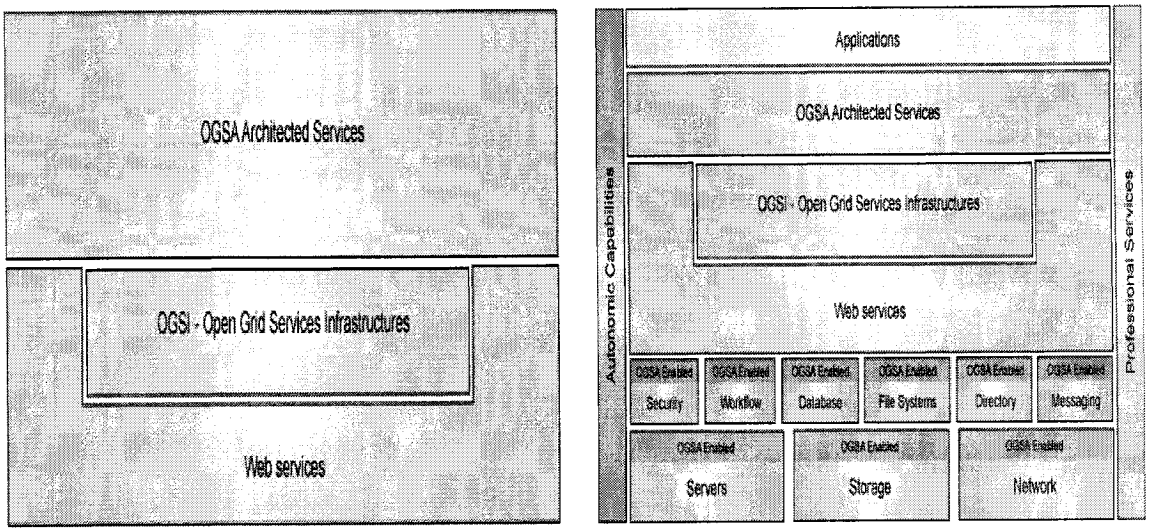

Figure 3 : OGSA architecture. [OGSA]

WSDA and OGSA are developed for fulfillment of almost same set of targets but they are both independent in nature. The main difference between OGSA and WSDA is OGSA is restricted to map a Grid Service Handle (GSH) to a Grid Service Reference (GSR) and in OGSA it is not obvious that every legal HTTP URL is a GSH; on the other hand in WSDA every legal HTTP link is a valid service link.

2.4.4 WSDA

[Hosc02a] proposes a unified and modular service discovery architecture for Grid computing, called Web Service Discovery Architecture (WSDA), which can be used in run time to discover and adapt appropriate remote services. WSDA facilitates an interoperable web service discovery layer by defining industry standard appropriate services, interfaces, and protocol bindings. The communication primitives facilitate service identification, retrieval of service description in a Grid computing environment. WSDA and OGSA [Fost02b] are proposed for fulfillment of almost same set of targets but they are both independent in nature. The main difference between OGSA and WSDA is OGSA is restricted to map a Grid Service Handle (GSH) to a Grid Service Reference

(GSR) and in OGSA it is not obvious that every legal HTTP URL is a GSH; on the other hand in WSDA every legal HTTP link is a valid service link.

2.4.5 Nimrod/G

[Buyy00a] proposes a "component based architectural design for Nimrod-G" which is implemented using different middleware services. Nimrod/G is a grid-enabled resource management service, which is an extended version of their previous "Nimrod" resource management architecture. This new Nimrod/G uses Globus Resource Allocation Manager (GRAM) to allocate the resource, Monitoring and Discovery (MDS) service to monitor and discover the resource, Grid Directory Information services for resource sharing. [Buyy00a] claimed that Nimrod/G could make good scheduling decisions.

2.4.6 RDF

According to [RDF], RDF stands for Resource Description Framework. This framework refers to describing and interchanging web metadata. As stated in [RDF-FAQ], "*Resource Description Framework is a universal format for data on the Web. Using a simple relational model, it allows structured and semi-structured data to be mixed, exported and shared across different applications. RDF data describe all sorts of things, and where XML schemas just describe documents, RDF and OWL schemas ("ontologies") talk about the actual things. This gives greater re-use. Where XML provides interoperability within one application (e.g. bank statements) using a given schema, RDF provides interoperability across applications (eg import your bank statements into your calendar). RDF metadata can be used in a variety of application areas. For example: in resource discovery to provide better search engine capabilities; in cataloging for describing the content and content relationships available at a*

particular Web site, page, or digital library; by intelligent software agents to facilitate knowledge sharing and exchange; in content rating; in describing collections of pages that represent a single logical "document"; for describing intellectual property rights of Web pages, and in many others. RDF with digital signatures will be key to building the "Web of Trust" for electronic commerce, collaboration, and other applications."

According to [RDF01], RDF is the core Metadata activity of W3C (World Wide Web Consortium). RDF can be used by various metadata applications of W3C; digital signatures, privacy preference management, PICS (Platform for Internet Content Selection) are some of them. The main concern of PICS is filter out unwanted contents from the web, e.g. filter out pornography, and other controversial contents.

According to [RDF], in RDF a **resource** is anything which has a Unified Resource Identifier (URI), a **property** is a resource which has name and which can be used as a property, a **statement** is a combination of resource, statement and value where resource, property, and value are known as "subject", "predicate", "object" respectively.

A simple RDF syntax [RDF]:

```
<rdf:Description about='http://www.textuality.com/RDF/Why-RDF.html'>
<Author>Tim Bray</Author>
<Home-Page rdf:resource='http://www.textuality.com' />
</rdf:Description>
```

RDF can be used in HTML[RDF-FAQ] :

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
 <rdf:Description rdf:about="http://doc"
    dc:creator="Joe Smith"
    dc:title="My document"
    dc:description="Joe's ramblings about his summer vacation."
    dc:date="1999-09-10" />
</rdf:RDF>
```

2.4.7 UDDI

According to [UDDI00], UDDI is a specification for distributed web-based information registers for web services. UDDI is a specification to publish and discover information of "Web services". According to this technical paper, Web service is an internet-based service, which can provide a specific service to another company or a software program to complete a particular task. XML based UDDI business registration service publishes information about a service for other interested party though "white pages", "yellow pages" and "green pages" components.

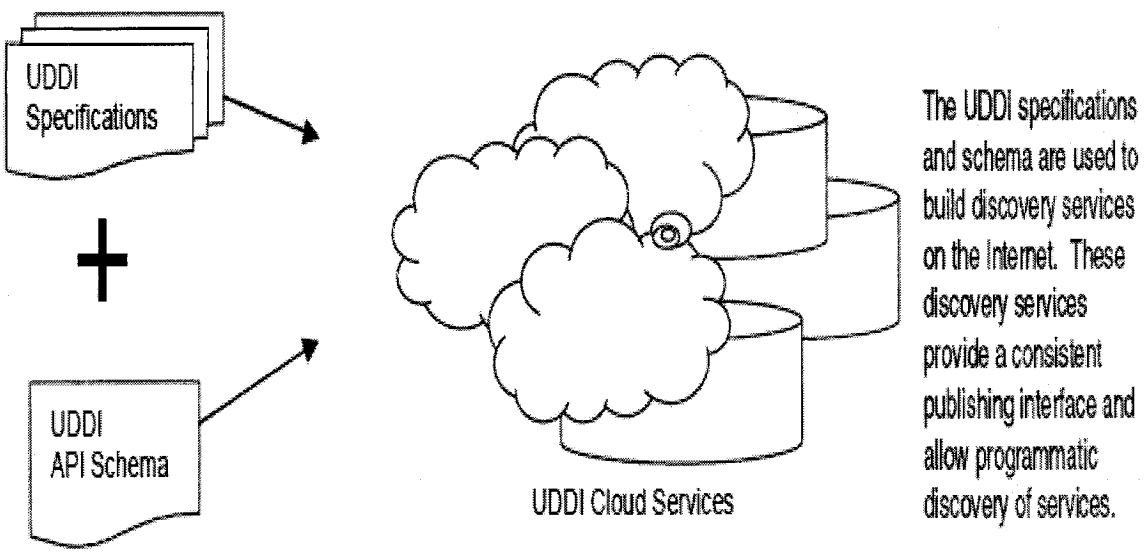

Figure 4 : High level view of UDDI. [UDDI00]

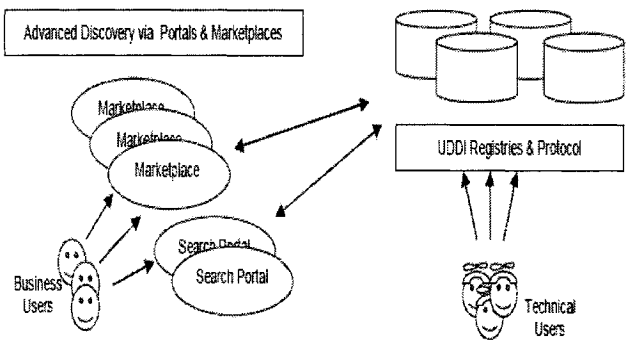

Figure 5: UDDI architecture at a glance [UDDI00]

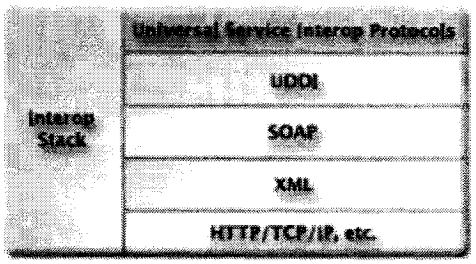

Figure 6: High level architecture view of UDDI [SUNUDDI]

According to [UDDI], The UDDI uses Worldwide Web Consortium (W3C) and Internet Engineering Task Force (IETF) standards mainly XML, and HTTP and Domain Name System (DNS) protocols. Cross platform programming capabilities are added to UDDI using Simple Object Access Protocol (SOAP). The UDDI protocol is mainly used for finding web services quickly and efficiently.

Currently UDDI is a widely used resource discovery mechanism in distributed network. IBM, Microsoft, and Sun Microsystems have been using UDDI for discovering the services. IBM offers a UDDI service called "IBM UDDI Registry" [IBMUDDI] service, where service provider can register his/her business and services, on the other hand a consumer can search that service thorough a web portal. In order to use this registry service one needs IBM userid and password. Currently this registry service is free to use. Microsoft also offers a free UDDI service called "UDDI Business Registry node" [MicroUDDI]. Similar to IBM approach, people can also register their businesses along with services to UDDI Business Registry node and interested consumer can find that specific service. Microsoft heavily uses UDDI in Windows 2003 Server, Office XP,

and Visual Studio .Net. "Sun also sees UDDI as an important project to establish a registry framework for business-to-business e-commerce"[SUNUDDI].

2.5 A General Comparison

Based on user friendliness, platform dependency, cpu/memory usage, cost (production cost/ management cost), possibility of integrating with other existing framework, publication of status of resources, following comparison table is established:

	Platform dependency	CPU/Memory friendliness	Cost	Possibility of integrating with other existing architecture	Status of the resources
Globus	Yes: Linux based	Under development	Very difficult to operate, so need help of technical expert	Very difficult	No
Web service	No	Yes	low	Yes	No
WSDA	No	N/A	High	Yes	No
OGSA	No	N/A	High	Yes	No
JXTA protocol	No	No	medium	Yes	No
UDDI	No	N/A	high	Yes	No

Table 1: Comparison among different frameworks/Architectures

2.6 Algorithms

2.6.1 Algorithms

The following algorithms have been reviewed in [Harc99], [Huan02], [Jun00], [Kutt01], [Kutt02], [Ludw02], [Law00], [Li02], [Mahe00].

2.6.1.1 Flooding Algorithm

According to [Jun00] and [Harc99], this algorithm is widely used by internet routers and where every node acts as a transmitter and receiver and every node tries to send every message to every node of its neighbour, a newly added new edge is not used for any communication, direct communication exists only in between initially existing set of neighbouring edges of the network. The required number of rounds of this algorithm is equivalent to the diameter of the graph. [Harc99] claims that this algorithm can be very slow if not started with a graph, which has small diameter.

2.6.1.2 The Swamping Algorithm

According to [Jun00] and [Harc99], swamping algorithm is similar to flooding algorithm except this algorithm allows a node to connect with all of its current neighbours, not only with the set of initial neighbours. [Harc99] suggests that the main advantage of this algorithm needs $O(\log n)$ rounds to converge to a complete graph and which is irrespective to the initial configuration. According to [Harc99], the disadvantage of this algorithm is communication complexity, this algorithm grows very quickly.

2.6.1.3 The Random Pointer Jump Algorithm

In this algorithm, in each round, each node contacts a random neighbour, and then this random neighbour sends all of its neighbours to the sender node. Finally sender neighbour and random neighbour's neighbours get merged. According to [Jun00] and [Harc99], a strongly connected graph with n nodes needs $\Omega(n)$ complexity time to converge to a complete graph.

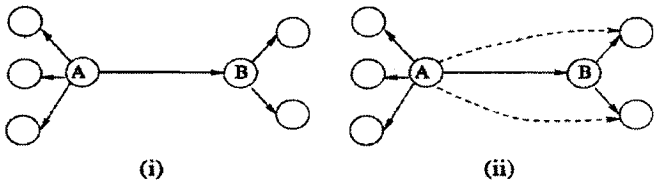

Figure 7: Random Pointer Jump ([Huan02], 1999, page: 232)

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

2.6.1.4 The Random Pointer Jump with Back Edge Algorithm

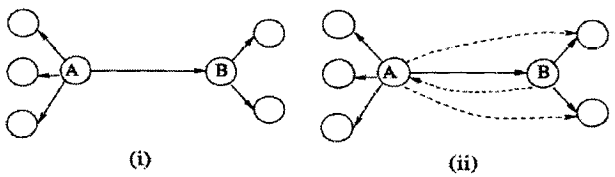

Figure 8: Random Pointer Jump with Back Edge ([Harc99], 1999, page: 233)

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

According to [Harc99] , this algorithm is almost identical to Random Pointer Jump algorithm except every time adding a back edge after performing the pointer jump.

2.6.1.5 Name Dropper Algorithm

This algorithm is proposed for querying resources in a weakly connected network where it is assumed that all machines already know each other. [Harc99]

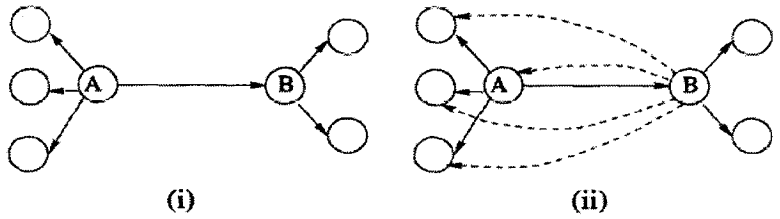

Figure 9: Name Dropper ([Harc99], 1999, page: 235)

(i) Before Name-Dropper. Node A chooses a random neighbour; here B is the chosen neighbour (ii) after one round of the Name-Dropper algorithm. A has passed to B all of its neighbours and B has added edges to these neighbours. In addition, B learns about A. The dashed lines indicate newly formed edges.

2.6.1.6 Distributed Awareness Algorithm

After analyzing different algorithms of [Harc99], [Jun00] proposes a new agent-based resource discovery algorithm called "Distributed Awareness Algorithm" and also they proposed a framework for dynamic assembly of agents based on this algorithm. Distributed awareness refers to a learning mechanism by which a node gets awareness about other nodes in a network. According to [Jun00], in this algorithm, each node has an awareness table and each node exchanges the information of this awareness table with other nodes. A typical awareness table entry contains location of the node (IP address), when last heard from the node, when last time the awareness information was sent to the node. The authors claimed that this agent based resource discovery system could provide better discovery services using its agents' autonomous behaviour.

2.6.1.7 Data Flooding Based Data Dissemination Algorithm

[Mahe00] proposed an algorithm called "Data Dissemination Algorithm" which follows swamping approach [Harc99] for message distribution. When a message comes to a node, that message gets validated. This validation process relies on three types of dissemination, universal awareness that permits all incoming messages, neighbourhood

awareness that allows messages from a certain distance, and distinctive awareness, which discards messages if it finds out that the less Grid potentiality at the local node in remote node, is less than that of the requestor node.

In [Mahe00], the authors also measured the performance of "universal awareness", "neighbourhood awareness", and "distinctive awareness" dissemination schemes. The authors claimed that universal approach is more expensive in terms of message complexity than that of neighbourhood and distinctive approach. The authors also claimed that this new class of dissemination could reduce the communication overhead during the resource discovery.

2.6.2 Classification of Different Algorithms

After reviewing all these algorithms we can classify them in following different groups:

Group Name	Algorithm
Centralized	Distributed Awareness Algorithm, Flooding Based Data Dissemination Algorithm
De-centralized	Flooding Algorithm, Swamping Algorithm, Random Pointer Jump Algorithm, Random Pointer Jump with Back Edge Algorithm, Name Dropper Algorithm

Table 2: Classification of algorithms

In Flooding Based Data Dissemination Algorithm, they concentrate on the "resource status database" in this algorithm, but our proposed model is little different than theirs. In

this research work we concentrate on the actual resource status which includes memory status, CPU status, available services, etc. On the other hand their "resource status" refers to the grid potentials, "*the Grid potential at a point in the Grid can be considered as the computing power that can be delivered to an application at that point on the Grid*".

2.7 Limitations of Previous Works

This above background reading suggests that no single approach and framework can solve all the problems and provide well engineered, platform independent, cpu/memory friendly, cost effective, and portable solution; some limited approaches or few frameworks provide the current status of the resources, but they are not well engineered or platform independent in nature. It would to be easier for an interested party to choose the particular resource, if the status of the resource is known or predicted before.

3 Status Based Resource Discovery Model

The ultimate goal of this system as described in Chapter 2 is to present a scalable, portable, easy to use resource status monitoring system in Computational Grids context. In this proposed model, all computers in a virtual organization should be able to update the status information database with their current status.

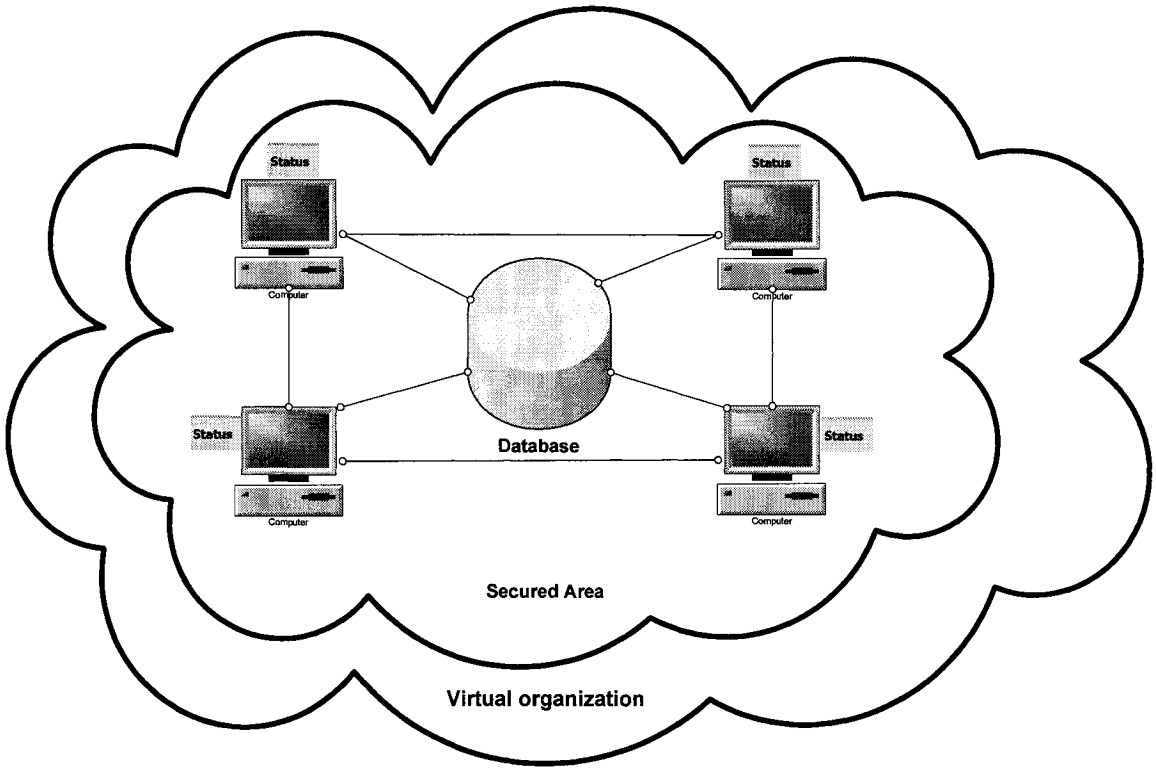

Figure 10: Status monitoring system in a virtual organization

3.1 Preliminary

First of all, we need to distinguish between two models: "resource status in Grid potential" [Mahe00] and resource status in our proposed model, although they seem similar from the out side. According to [Mahe00], "Grid Potential" refers to the computing power which includes, CPU speed, FLOP rating, sustained memory access rate, persistent disk access rate, which are almost static in nature. But in our proposed

model, we are interested to know the static information as well as the dynamic status of the resources. For example, total memory, and total power of the CPU are two main static resources in our model. In addition, we also retrieve the dynamic information of the resources which includes current CPU usage, current available memory, and current available services.

The work done by [Czaj01] is similar to our approach.[Czaj01] describes issue and solution of monitoring of services and resources; they implemented the Grid Information Service for Globus system, which is another platform dependent solution. In addition to that, in their work they totally ignored the existence of platforms other than UNIX based system. But in our work, we propose an architecture which works in multiple platforms.

3.2 Assumptions

In order to accomplish our goals we need to make following assumptions:

- The resources we are considering are mainly low end desktop computers
- They are connected to networks (such as the web) preferably with static IP.
- These resources are not used for completely personal use. As for example, watching movies, listening to music, running downloading software all the time (Kazaa, iMesh, Napster, etc).

3.3 The Proposed Model

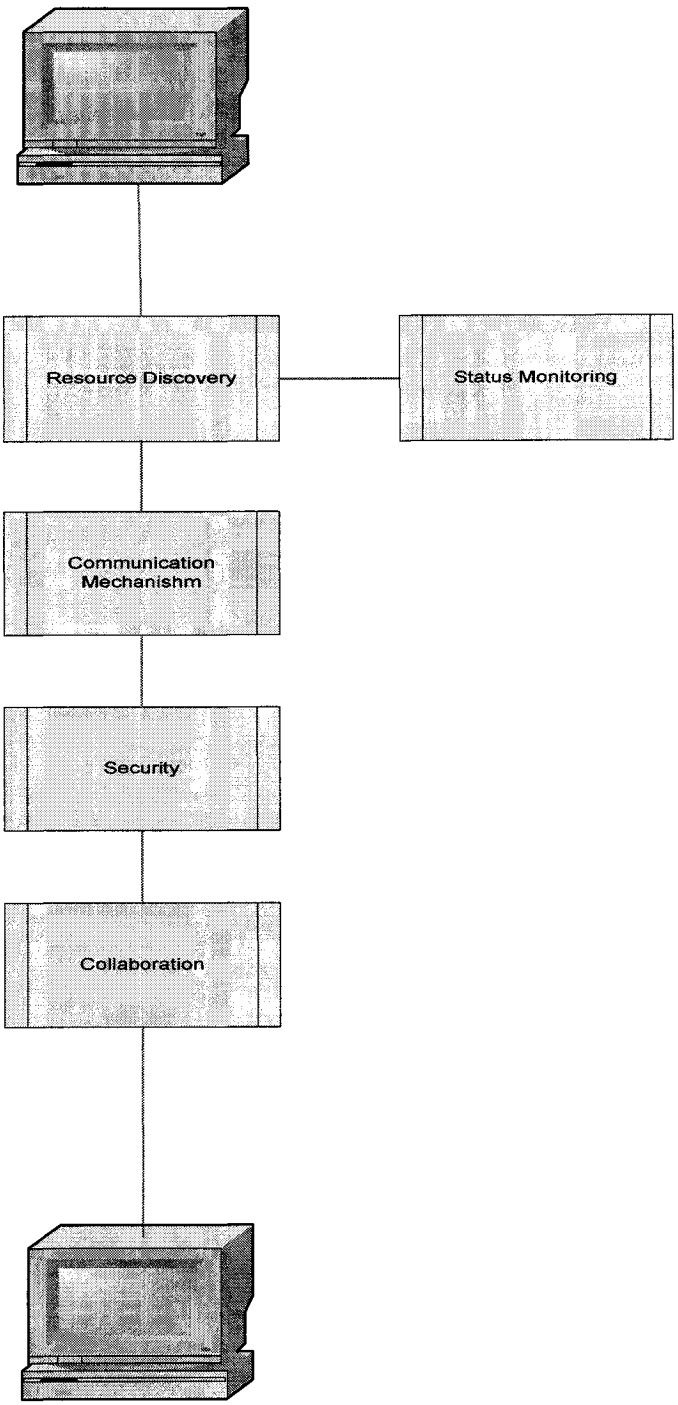

Figure 11: Possible place for Status Monitoring System in a network

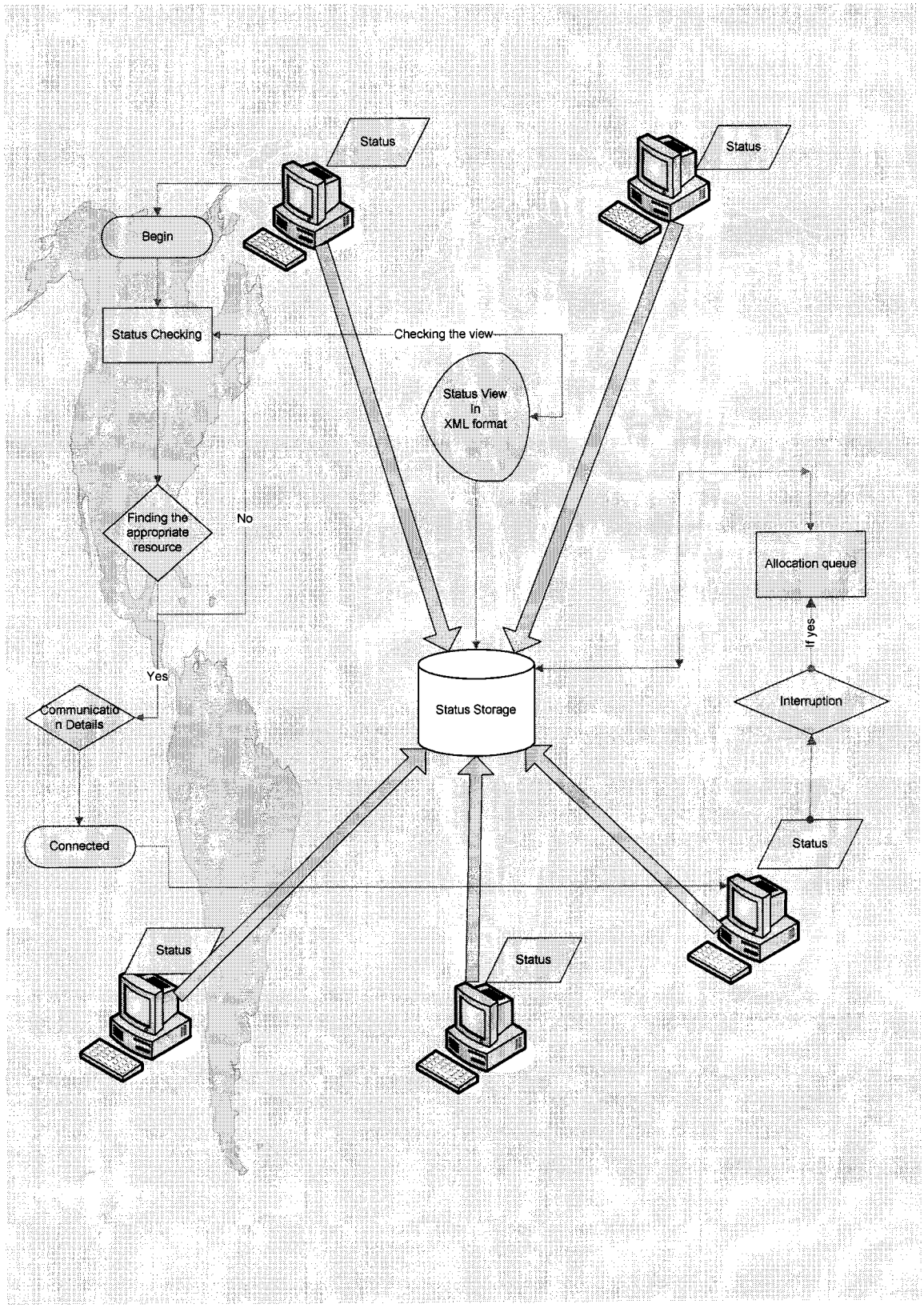

Figure 12: System architecture and flowchart

3.3.1 System Overview

In this thesis work, we propose a resource discovery infrastructure in the form of an automated status monitoring model. The model consists of two fundamental aspects, a portable data model and a set of executable monitoring components. Our approach adheres to principles of software design, is well structured and platform independent. The portable data model, which conveys the status of the resources, must be understandable by any application software, agent or scheduler on any platform. In turn, the monitors must be able to acquire necessary status information from various, diverse systems and maintain the data model.

There are four main components available in this system: central status monitoring repository, interested resource provider and consumer, network connection, and self – executable resource monitoring and reporting agent.

The main target of our research work is to present a resource discovery architecture based on monitoring capabilities.

3.3.2 Implementation Details

3.3.2.1 Tools being used

In order to implement this proposed model, the following components, tools, operating systems, programming languages, frameworks are used.

- OS Environment:
    - Linux (Red Hat 7.3)
    - Windows (Windows XP)
- Programming Language
    - C for Linux

- C# for windows
- PHP for the web

- Database
  - MySQL Server (v 1.4)
  - MySQL ODBC Driver for windows
  - Pro*C
  - Windows Management Instrumentation (WMI)
  - Windows Management Instrumentation Query Language (WQL) to query the status in Windows environment

- Tools
  - Existing Linux Tools: top, vmstat, fd
  - Existing Windows Tool: perfmon

- Publication Tools:
  - XML
  - PHP

In this section we are not going to discuss about all of them, only WMI, WQL, Perfmon, Linux system applications are discussed below:

WMI:

As stated in [WMI]:

"The WMI infrastructure is a Microsoft® Windows® operating system component that moves and stores information about managed objects. The WMI infrastructure is made of two components: the Windows Management service, and the WMI repository. The Windows Management service acts as an intermediary between the providers,

management applications, and the WMI repository, placing information from a provider into the WMI repository. The Windows Management service also accesses the WMI repository in response to queries and instructions from management applications. Finally, the Windows Management service can pass information directly between a provider and a management application. In contrast, the WMI repository acts as a storage area for information passed in by the various providers.

Windows Management Instrumentation (WMI) is a component of the Microsoft® Windows® operating system and is the Microsoft implementation of Web-Based Enterprise Management (WBEM), which is an industry initiative to develop a standard technology for accessing management information in an enterprise environment. Windows Management Instrumentation (WMI) provides access to enterprise control and WMI provides security measures to restrict access. The WMI infrastructure provides security by using identification, impersonation, and namespace access. The WMI infrastructure provides security by positively identifying a user. After identifying a user, the WMI infrastructure validates user credentials before a user can log on to WMI."

WQL:

According to [WQL], WQL stands for Windows Management Instrumentation Query Language, a subset of ANSI SQL (American National Standards Institute Structured Query Language). With the minor semantic changes from ANSI SQL, it is used to support WMI.

A complete list of supported SQL keywords can be found at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/sql_for_wmi.asp

Perfmon:

"Perfmon"[Perfmon] is a built-in Windows Operating System tool which is used to monitor the performance of the system, mainly cache, memory, objects, paging file, physical disk, process, processor, server, system, thread.

Linux System Applications:

We use two Linux applications in order to gather system information in a Linux environment: *"top"*, *"fd"*, and *"vmstat"*. According to [LinuxCommand], *"top"* command displays the system information such as average load, CPU usages, uptime and processes. *"vmstat"* displays information about processes, memory, paging, block IO, traps, and CPU activity. *"fd"* displays the storage and partition information.

3.3.2.2 Design

We have already mentioned above that four main components are required to construct this system: central status monitoring repository, interested resource provider and consumer, network connection, self – executable resource monitoring and reporting agent.

Central status monitoring repository: This is one of the man components of this system. The central status monitoring repository is a central location where the current status of the resources are stored in order to help the requestor's agent to choose the appropriate resources according to the needs of the requestor.

Resource provider and consumer: In this system, resource providers play the main role of resource sharing. A resource provider is a volunteer entity who wants to share his/her resources with the other members at the network. On the other hand, a consumer wants to

use the resources from a resource service provider. In any resource sharing-distributed environment a resource provider can be a consumer and vice versa.

Network connection: In section 3.3, we have already assumed that all members of the systems are connected through a network; preferably members have the static IP address in order to maintain the connection.

Resource monitoring and reporting agent: A resource monitoring agent is responsible for monitoring the system. A reporting agent is responsible for updating the central repository with the latest status of the local system. It also updates the queue manager with the situation of the system, if any interruption occurs. For example, when service provider's resources provide the consumer with the service, if service provider's computer needs to do something for its own, the reporting agent will notify the queue manager and queue manager keeps the current progress of the work, and forwards the request to another service provider or connect the same provider after a certain time.

3.3.2.3 Implementation

The implementation of this architecture has been designed for multiple operating systems, mainly Windows 2000, and Windows XP operating system and Linux based operating system which is tested in RedHat 7.3 Kernel version 2.4.18-3. For the windows based environment the implementation has been achieved by Net's "C#" programming language with the support of WMI and WQL. For the Linux based operating systems the implementation has been completed by "C" language with the support of existing Linux applications (***top, vmstat, free, fd***). Open source database solution "MySQL v. 4.0.18-standard" is used to achieve the central status monitoring repository support.

The client application has been tested on Intel Pentium, PII, PIII, PIV processors with the support of at least 64Mbs of RAM. The database server has been configured on an Intel PIV machine with 256 Mbs of RAM support.

The network between different machines is provided by regular high speed internet connection.

During the operation, behaviour of the members' machines is closely monitored. Especially, the CPU and memory consumption by the client application are monitored. On average 5.676377% CPU consumption and 11.84928 Mbs of RAM consumption are observed. In both of the Windows and Linux environments, the client program takes more RAM in its starting phase than the regular phase. Since, the monitoring program monitors the status of the resource machine periodically, thread management technique is used to make the activities idle when there is no need to monitor.

This architecture also has on demand a reporting mechanism. For example, if the owner of the resources needs the resources for himself/herself during an execution of an operation, the reporting agent captures this exception and sends the status of the job to scheduler and the scheduler may store this incomplete work in the queue and assigns this job to another available resource(s) if needed.

4 Experimentation and Result

In this chapter two experiments are designed and executed to determine the feasibility and the effectiveness of this proposed architecture. These experiments have been conducted on Windows environment.

The experiments are conducted based upon two different scenarios. Case 1 presents how to monitor the system and Case 2 presents how to search for a resource.

In section 4.1 case 1 is presented. In section 4.2 Case 2 is presented. In section 4.3, test results are summarized. In section, 4.4 technical difficulties are discussed.

4.1 Case 1

The status monitoring application is easy to run. Anybody who is a member of a networked community can download the application based on the operating system. In this case, when the application starts running, it captures the status of the machine in a timely fashion; in this case the interval is 5 minutes. That means, in every 5 minutes, it takes a snapshot of the resource, mainly the CPU utilization and the memory usage. After getting the status, it passes this information to a central repository system. The process goes to "sleep" when there is no work do. The central repository system stores the data and when an interested party needs the resource for a task, the application agent can query this status database table for a matching result. After finding a matching result, the application can send a request to the service provider computer and finish the job. But, if any interruption occurs during the execution, "Reporting Agent" can contact with the "Scheduler Queue Agent" and the Queue agent may store this information and pass the request to another service provider.

```
C:\thesis\C>project_thesis
December 2, 2004, 9:51:59 PM
Sucesful
UPDATE LOW_PRIORITY PROFILE SET DATE_TIME ='December 2, 2004, 9:51:59 PM((GMT-05
:00) Eastern Time (US & Canada) )', FREE_MEMORY ='34', FREE_CPU ='79.66666' WHER
E MEMBER_ID = 1
(GMT-05:00) Eastern Time (US & Canada)  Intel Pentium III processor 535 MHZ
Microsoft Windows XP Professional       318     34      79.66666%       24.57.48
.220
```

Figure 13: Monitoring agent is in the action

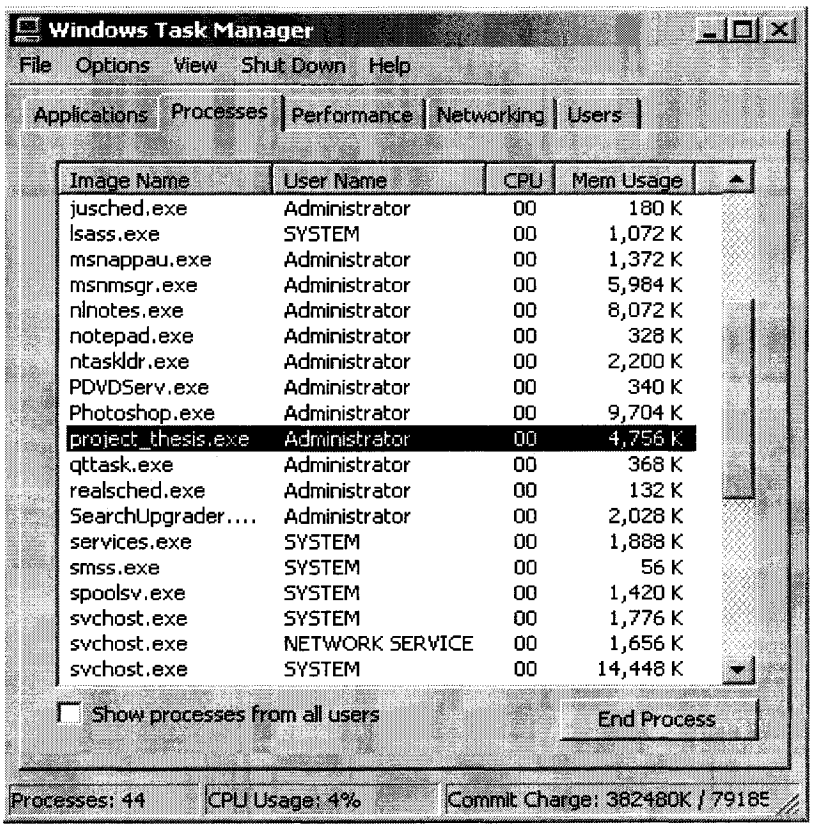

Figure 14: Monitoring agent is in its sleeping state

The following table has been constructed with real data over 6 hours period of time span.

Here, we try to capture the behaviour of the application and service provider's machine.

Processor: Intel Pentium III processor 535 MHZ
Total Memory: 318 Mega bytes
Operating System: Windows XP

Table 3: CPU and Memory Consumption of the system and the application

Time	CPU usage by the application (%)	Memory usage by the application	Free CPU (%)	FREE Memory
12:31:35 AM	10.67	21 Mbs	78.33	158 Mbs
12:36:21 AM	4	10 Mbs	89	129 Mbs
12:41:22 AM	5	12 Mbs	92.61	111 Mbs
12:46:22 AM	6	4.9 Mbs	91.98	101 Mbs
12:51:22 AM	9	5.2 Mbs	91.99	100 Mbs
12:56:22 AM	7	10 Mbs	91.69	94 Mbs
1:01:23 AM	5	12 Mbs	92.22	86 Mbs
1:06:23 AM	4	6 Mbs	91.33	85 Mbs
1:11:23 AM	8	5.2 Mbs	90.77	86 Mbs
1:16:24 AM	6	15 Mbs	91.66	86 Mbs
1:21:24 AM	4	23 Mbs	91.93	87 Mbs
1:26:24 AM	9	22 Mbs	91.91	86 Mbs
1:31:24 AM	4	10 Mbs	91.64	87 Mbs
1:36:25 AM	3	12 Mbs	91.33	85 Mbs
1:41:25 AM	4	4.9 Mbs	91.26	84 Mbs
1:46:25 AM	7	5.2 Mbs	91.86	84 Mbs
1:51:26 AM	8	10 Mbs	91.67	84 Mbs
1:56:26 AM	4	12 Mbs	91.67	84 Mbs
2:01:26 AM	5	6 Mbs	91.63	84 Mbs
2:06:26 AM	7	5.2 Mbs	91.62	84 Mbs
2:11:27 AM	8	15 Mbs	91.60	84 Mbs
2:16:27 AM	4	23 Mbs	91.56	84 Mbs
2:21:27 AM	8	21 Mbs	91.66	84 Mbs
2:26:27 AM	3	10 Mbs	91.98	84 Mbs
2:31:28 AM	4	12 Mbs	91.19	84 Mbs
2:36:28 AM	9	4.9 Mbs	91.89	84 Mbs
2:41:28 AM	8	5.2 Mbs	91.32	78 Mbs
2:46:28 AM	8	19 Mbs	91.99	73 Mbs
2:51:29 AM	6	12 Mbs	91.65	72 Mbs
2:56:29 AM	6	6 Mbs	91.79	72 Mbs
3:01:29 AM	4	5.2 Mbs	90.66	72 Mbs
3:06:29 AM	5	15 Mbs	91.69	72 Mbs

3:11:30 AM	5	23 Mbs	91.54	72 Mbs
3:16:30 AM	3	21 Mbs	91.74	72 Mbs
3:21:30 AM	6	10 Mbs	89.59	72 Mbs
3:26:30 AM	7	12 Mbs	90.37	72 Mbs
3:31:31 AM	2	4.9 Mbs	91.59	72 Mbs
3:36:31 AM	3	5.2 Mbs	91.53	72 Mbs
3:41:31 AM	6	10 Mbs	91.69	69 Mbs
3:46:32 AM	2	12 Mbs	90.89	72 Mbs
3:51:32 AM	4	6 Mbs	92.34	72 Mbs
3:56:32 AM	6	15 Mbs	93.48	72 Mbs
4:01:32 AM	5	15 Mbs	94.94	72 Mbs
4:06:33 AM	5	23 Mbs	91.99	72 Mbs
4:11:33 AM	3	21 Mbs	90.65	71 Mbs
4:16:33 AM	5	10 Mbs	91.34	71 Mbs
4:21:33 AM	6	12 Mbs	91.91	71 Mbs
4:26:34 AM	3	4.9 Mbs	91.39	71 Mbs
4:31:34 AM	4	5.2 Mbs	90.63	71 Mbs
4:36:34 AM	6	15 Mbs	91.25	71 Mbs
4:41:34 AM	5	12 Mbs	91.39	72 Mbs
4:46:35 AM	8	6 Mbs	91.57	72 Mbs
4:51:35 AM	4	5.2 Mbs	91.35	72 Mbs
4:56:35 AM	5	15 Mbs	91.51	72 Mbs
5:01:35 AM	7	23 Mbs	91.97	72 Mbs
5:06:36 AM	5	21 Mbs	91.35	72 Mbs
5:11:36 AM	7	10 Mbs	89.99	72 Mbs
5:16:36 AM	7	12 Mbs	89.99	72 Mbs
5:21:36 AM	5	4.9 Mbs	90.39	71 Mbs
5:26:37 AM	7	5.2 Mbs	91.35	72 Mbs
5:31:37 AM	5	10 Mbs	91.95	73 Mbs
5:36:37 AM	6	12 Mbs	91.38	72 Mbs
5:41:37 AM	8	6 Mbs	91.11	72 Mbs
5:46:38 AM	8	5.2 Mbs	91.56	72 Mbs
5:51:38 AM	8	15 Mbs	91.73	72 Mbs
5:56:38 AM	6	23 Mbs	91.19	71 Mbs
6:01:38 AM	4	12 Mbs	91.78	72 Mbs
6:06:39 AM	9	17 Mbs	91.79	73 Mbs

6:11:39 AM	4	18 Mbs	91.71	73 Mbs

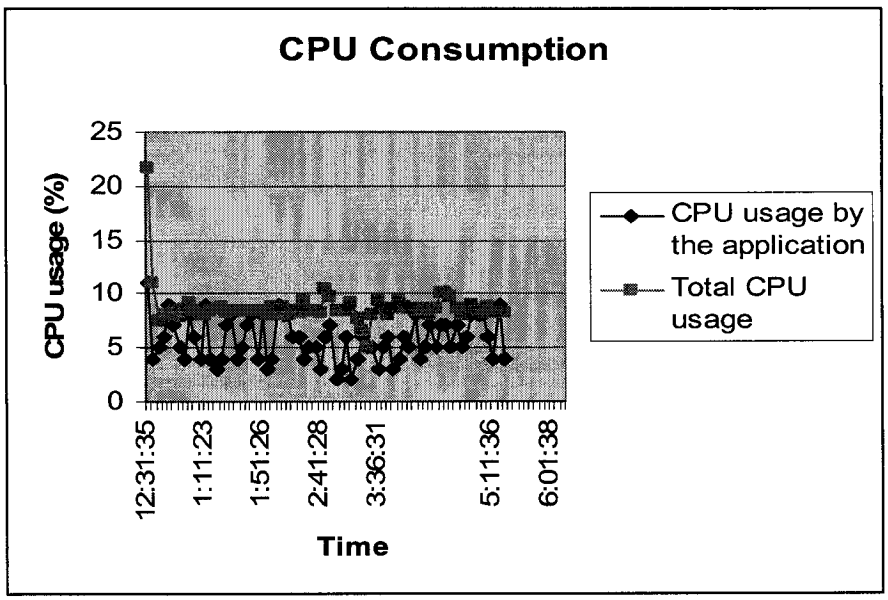

Figure 15: Comparison of the CPU usage between the total CPU usage and CPU usage by the application

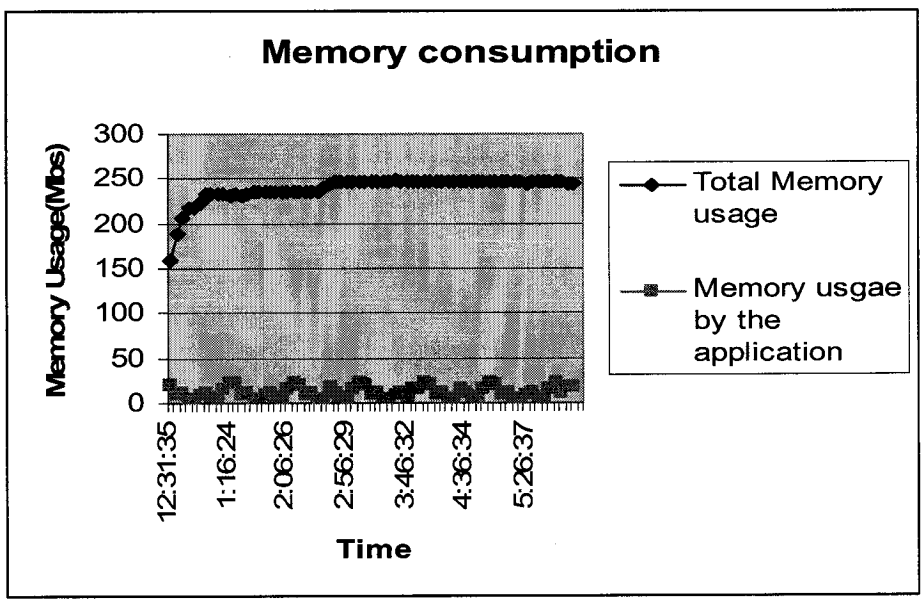

Figure 16: Comparison of the memory usage between total memory usage and memory usage by the application

4.2 Case 2

In this section, we present how to search for a resource and the final matching result. Since, the definition of "PROFILE" table is open to the public, anybody can search for the resources in the repository system or can use the web interface to search for the resources.

```
mysql> desc PROFILE;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| MEMBER_ID    | int(11)      |      | PRI | 0       |       |
| MEMBER_IP    | varchar(30)  | YES  | MUL | NULL    |       |
| MEMBER_NAME  | varchar(100) | YES  |     | NULL    |       |
| SERVICE_LIST | text         | YES  |     | NULL    |       |
| DATE_TIME    | varchar(100) | YES  |     | NULL    |       |
| TIME_ZONE    | varchar(100) | YES  |     | NULL    |       |
| CPU          | varchar(100) | YES  |     | NULL    |       |
| OS           | varchar(100) | YES  |     | NULL    |       |
| TOTAL_MEMORY | varchar(20)  | YES  |     | NULL    |       |
| TOTAL_SWAP   | varchar(20)  | YES  |     | NULL    |       |
| FREE_MEMORY  | varchar(20)  | YES  |     | NULL    |       |
| FREE_SWAP    | varchar(20)  | YES  |     | NULL    |       |
| FREE_CPU     | varchar(20)  | YES  |     | NULL    |       |
| DISK         | text         | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
14 rows in set (0.00 sec)
```

Figure 17: Definition of Profile table

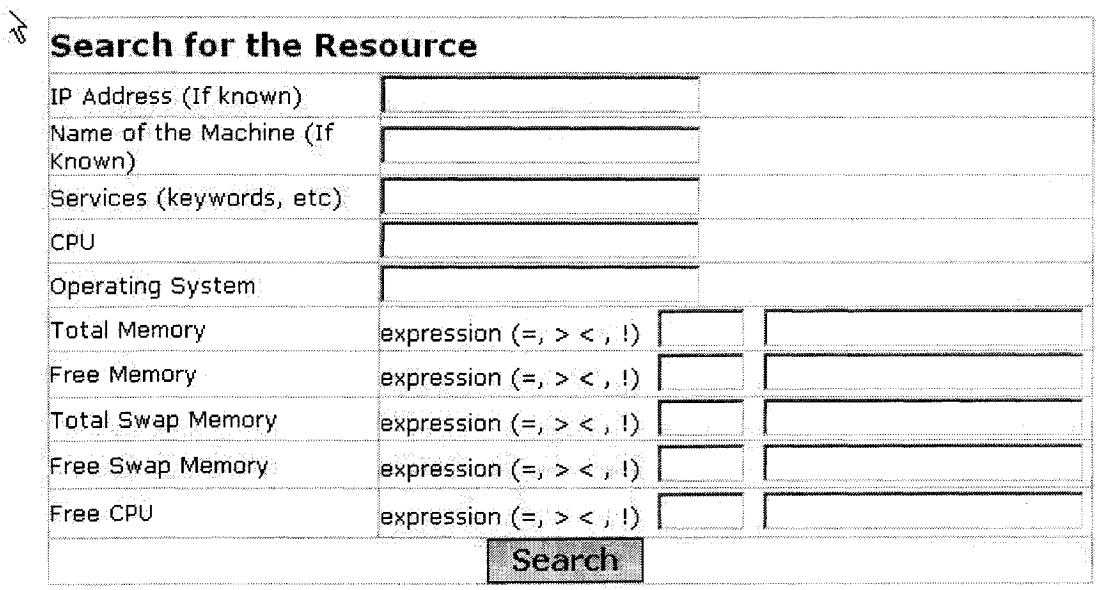

Figure 18: Web-based search interface

```
<?xml version="1.0" encoding="UTF-8" ?>
- <resources>
    <resources_found>2</resources_found>
  - <resource>
      <name>kent2</name>
      <ip>137.207.234.170</ip>
      <service>Bubble sort in C,Bubble sort in Java</service>
      <cpu>GenuineIntelPentium III (Coppermine) 797.438 MHz</cpu>
      <operating_system>Red Hat Linux 7.3 2.96-110</operating_system>
      <storage>Name : C: Total Disk Size in Mbytes:1941 Free Disk Size in Mbytes:226 Name : D: Total Disk Size in
        Mbytes:11613 Free Disk Size in Mbytes:4147 Name : E: Total Disk Size in Mbytes:5032 Free Disk Size in
        Mbytes:3711 Name : F: Total Disk Size in Mbytes:5017 Free Disk Size in Mbytes:1138 Name : G: Total Disk
        Size in Mbytes:4988</storage>
      <total_memory_mb>249.664062</total_memory_mb>
      <free_memory_mb>10.535156</free_memory_mb>
      <total_swap_memory_mb>996.207031</total_swap_memory_mb>
      <free_swap_memory_mb />
      <free_cpu_percentage>99.699997</free_cpu_percentage>
      <time_zone />
    </resource>
```

Figure 19: Query result in XML format after passing the query string "select * from PROFILE"

4.3 Results

From this result we can see that resource consumption by the application is stable, and it uses a reasonable amount of resources. So, we can conclude that from the CPU and memory consumption point of view, running this application is not expensive. Since, we keep the definition of "PROFILE" table for the public, searching for a specific resource is well- engineered in nature.

4.4 Technical difficulties

We have faced the following technical difficulties during the implementation.

4.4.1 Memory usage:

Most of the time, the application consumes more resources than at the beginning of the execution of the client's application. For example, in a general cases, the application takes less than 10 Mbs of RAM (lowest 255 Kbs was observed) to continue the process, but at the beginning the highest RAM consumption was observed 25 Mbs. But over the life span of the application the RAM consumption becomes lesser up to a certain limit.

4.4.2 Instability of MySQL Connector for Windows Platform:

In order to implement this effort, MySQL database server was deployed in a Linux box. A MySQL database connector or ODBC connector (ODBC 3.51 has been used) was needed to create a bridge between Windows based application and the MySQL server. Connection drop out was observed in some instances. We also use MySQL client for connecting the Linux based application and the database server; so far, no drop out has been recorded.

4.4.3 Interaction between a Scheduler and Reporting Agent

In our proposed model, we propose a reporting agent, which can monitor any interruption during an execution and can report to the scheduler if needed. In this case, reporting agent is not implemented programmatically, so there is no real interaction between the proposed reporting agent and the scheduler.

5 Summary and Future Work

5.1 Summary

In this thesis, we dealt with the issues related to resource monitoring and discovery in Computational Grids. After reviewing the different existing Grid architectures, we have found that grid computing can be more meaningful with a proper resource discovery architecture based on the current status of the systems.

In this thesis work, we designed, implemented and tested a discovery infrastructure in the form of an automated status monitoring model. The model consists of two main fundamental aspects, portable data model and a set of executable monitoring components. Our approach adheres to principles of software design, is well structured and platform independent. The portable data must be understandable by any application software, agent or scheduler on any platform. In turn, the monitors must be able to acquire necessary status information from various, diverse systems and maintain the data model. We developed appropriate interfaces that provide straightforward connectivity between our infrastructure and other Grid middleware components being developed elsewhere.

This research work aims to tackle the issue of resource discovery along with the status of the system on any platform, any operating system.

In short, the main contributions of this research work are given below:

a) Currently we can collect different status information of a resource provider, which includes storage capacity, memory capacity, SWAP information, time zone, available software, operating system information, CPU usage information. This list can be extended in the future.

b) Well engineered heterogeneous resource monitoring architecture

c) In this system, the resource provider will provide the information about resources in a timely manner; thus service provider will not be interrupted when he/will is busy.

d) CPU/Memory friendly solution: the application consumes less CPU and less memory

f) The output of this system can be easily integrated with other system. For example, another student of our research group has done a research on scheduler based on genetics algorithm. The input of that system is IP address and free CPU usage of a computer. Currently, static input is used for that system, but we can easily supply that input from our program.

5.2 Future Work

In the course of conducting this research effort we came across some issues that we believe make for good research opportunities to implement the whole architecture.

5.2.1 Communication among Different Virtual Organizations

In this thesis, we have implemented the resource discovery mechanism for a single virtual organization. But it is possible to connect multiple virtual organizations. To achieve this target, we can use one of the algorithms we have described in section 2.6.1 such as the Data Flooding Based Data Dissemination Algorithm as a communication model. For the decision making purpose we can use one of the approaches among Peer-to-Peer Approach, De-Centralized approach, Agent-Based Approach, Ontology Description-Based Approach, Routing Transferring Model-Based Approach, Parameter-Based Approach, Quality of Service (QoS)-based Approach, and Request Forwarding Approach, which we have already discussed in the section 2.3.

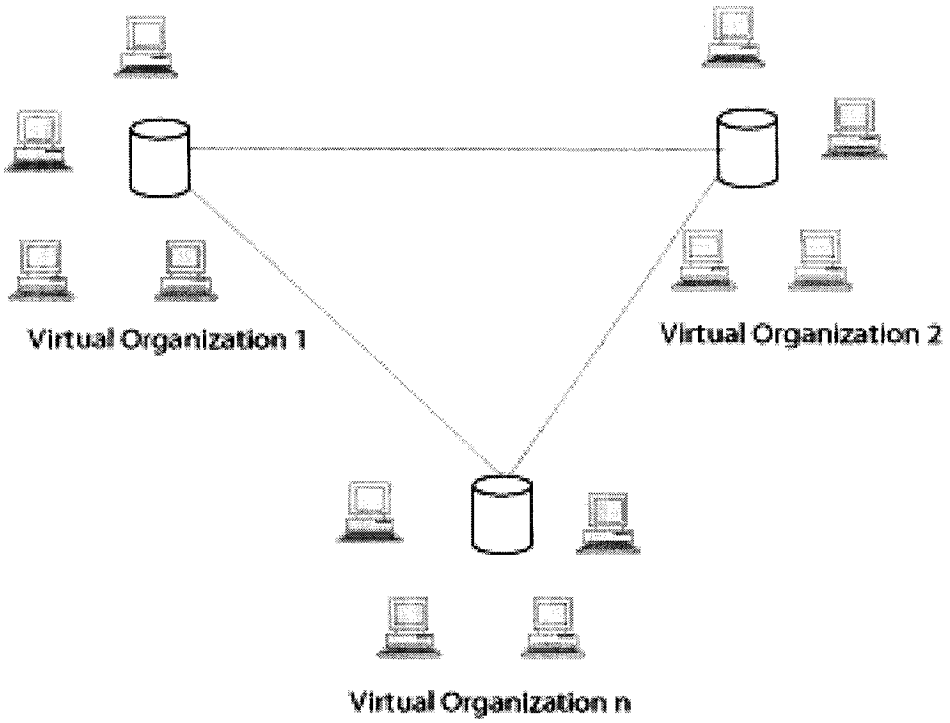

Figure 20: Resource Discovery in multiple virtual organizations

5.2.2 Real Time Interruption Monitoring Tool

We have identified a need for having a real time Interruption Monitoring tool which could be useful to capture any interruption during any execution of a task and report to the scheduler or to a job allocating agent. We believe that it could be a good research opportunity and using this tool, the execution of a task could be completed successfully.

5.2.3 Behaviour of Resources

We believe that it is possible to select a resource based on the colleted data along with other criteria for a period of time, say 1 Month or 2 Months. Since, activities over a computer are dependent on the habits of its user's usage. For example, if the user likes to work during the day, his/her computer is almost free in the night. So, we can isolate this resource before making any query.

References

1. [Alle01] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, J. Shalf. **The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment**. *A technical Report: TR-2001-28*,University of Chicago, 2001

2. [Alle99] J.Allen, M. Mealling. **The Architecture of the Common Indexing Protocol (CIP)**. *RFC-2651*, 1999

3. [Buyy00a] R. Buyya, D. Abramson, J. Giddy. **Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid.** *Proc. of the HPC ASIA'2000, the 4th International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, IEEE Computer Society Press*, pp: 283-289, USA, 2000

4. [Buyy00b] R. Buyya, S. Chapin, D. DiNucci. **Architectural Models for Resource Management in the Grid**. *The First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Springer Verlag LNCS Series*, pp: 18-35, 2000

5. [Chan02] A.Chander, S. Dawson, P. Lincoln, D. Stringer-Calvert. **NEVRLATE: scalable resource discovery**. *Cluster Computing and the Grid 2nd IEEE/ACM International Symposium CCGRID2002*, pp: 382, 2002

6. [Czaj01] K. Czajkowski, S. Fitzgerald, I. Foster, C.Kesselman. **Grid Information Services for Distributed Resource Sharing**. *Proc. of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10),* pp: 181, IEEE Press, 2001

7. [Czaj98] K.Czajkowski, I. Foster, N. Karonis, C. Kesselm an, S. Martin, W.Smith, S.Tuecke. **A resource management architecture for metacomputing systems**. *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pp: 62–82.Springer-Verlag LNCS 1459, 1998

8. [Fost01] I. Foster, C. Kesselman, S. Tuecke. **The Anatomy of the Grid - Enabling Scalable Virtual Organizations**. *Proc. of First IEEE/ACM International Symposium on Cluster Computing and the Grid,* pp: 6 - 7, 2001

9. [Fost02a] I. Foster**. The Grid: A New Infrastructure for 21st Century Science.** *Physics Today, 55(2), pp: 42-47,* 2002.

10.[Fost02b] I.Foster, C. Kesselman, J. Nick, S. Tuecke. **The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.** *Open Grid Service Infrastructure WG, Global Grid Forum,* 2002

11.[Fost02c] I. Foster. **What is the Grid? A Three Point Checklist**. GRIDToday, July 20, 2002

12.[Fost97] I. Foster, C. Kesselman. **Globus: A metacomputing infrastructure toolkit.** *International Journal of Supercomputer Applications*,11(2), pp: 115-128, 1997

13.[Fost98] I Foster, C. Kesselman (eds.). **Computational Grids", *The Grid: Blueprint for a New Computing Infrastructure.*** *Morgan-Kaufman*, San Fransisco,1998

14.[Globus] **Globus Architecture.** Retrieved November 10, 2004, from http://www.globus.org/presentations/cactus/sld003.htm

15.[Harc99] M. Harchol-Balter, T. Leighton, D. Lewin. **Resource Discovery in Distributed Networks.** *18th ACM Symposium on Principles of Distributed Computing*, pp: 229-237, 1999

16.[Hosc02a] W. Hoschek. **The Web Service Discovery Architecture.** *Proc. of the 2002 ACM/IEEE conference on Supercomputing*,2002

17.[Hosc02b] W. Hoschek. **A Unified Peer-to-Peer Database Framework for Scalable Service and Resource Discovery.** *Proc. of Third International Workshop on Grid Computing: GRID 2002,* Baltimore, MD. , pp: 126-144, Springer, 2002

18.[Huan02] Y. Huang, N. Venkatasubramanian. **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

19.[Iamn01] A. Iamnitchi and I. Foster. **On Fully Decentralized Resource Discovery in Grid Environments.** *IEEE International Workshop on Grid Computing,* Denver, CO, 2001.

20.[Iamn02a] A.Iamnitchi. **Resource Discovery In Large-Scale Distributed Environments.** *Doctoral thesis proposal,* University of Chicago, 2002

21.[Iamn02b] A. Iamnitchi, I. Foster, D. Nurmi. **A Peer-to-Peer Approach to Resource Discovery in Grid Environments.** *Proc. of the 11th Symposium on High Performance Distributed Computing,* Edinburgh, UK, 2002

22.[Iamn02c] A Iamnitchi, I. Foster, Daniel C. Nurmi. **A Peer-to-Peer Approach to Resource Discovery in Grid Environments.** *Proc. of the 11th Symposium on High Performance Distributed Computing,* Edinburgh, UK, 2002

23.[IBMUDDI] **Web Services and UDDI.** Retrieved December 1, 2004, from http://www-306.ibm.com/software/solutions/webservices/uddi/

24.[Jun00] K. Jun, L. Bolon, K. Palacz, D. Marinescu. **Agent-based resource discovery**. *Proc. of IEEE Heterogeneous Computing Workshop, 2000. (HCW 2000),* pp: 43 -52, 2000

25.[JXTA] **JXTA v2.0 Protocols Specification.** Retrieved December 1, 2004, from http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html

26.[Krau00] K. Krauter, R. Buyyaa, and M. Maheswaran. **A Taxonomy and Survey of Grid Resource Management Systems**. *Technical Report: Manitoba University (Canada) and Monash University (Australia),* 2000

27.[Kutt01] S. Kutten , D. Peleg , U. Vishkin. **Deterministic resource discovery in distributed networks**. *Proc. of the thirteenth annual ACM symposium on Parallel algorithms and architectures* , pp: 77-83 2001

28.[Kutt02] S. Kutten, D. Peleg. **Asynchronous resource discovery in peer to peer networks.** *Proc. of 21st IEEE Symposium on Reliable Distributed Systems,* pp: 224 -231, 2002

29.[Lars01] R. Larson. **Distributed resource discovery: using z39.50 to build cross-domain information servers**. *Proc. of the first ACM/IEEE-CS joint conference on Digital libraries,* pp: 52-53, 2001

30.[Law00] C. Law , Kai-Yeung Siu. **An O(log n) randomized resource discovery algorithm.** *The14th International Symposium on Distributed Computing, Technical Report,* Technical University of Madrid, pp: 5-8, Oct. 2000

31.[Li02] W. Li, Z. Xu, F. Dong, J. Zhang. **Grid Resource Discovery Based on a Routing-Transferring Model**. *Proc. of Third International Workshop on Grid Computing: GRID 2002,* Baltimore, MD. , pp:145-156, Springer, 2002

32.[LinuxCommand] **Linux Shell Command Library.** Retrieved December 1, 2004, from http://www.linuxforum.com/shell/top/43-17.php

33.[Ludw02] S. Ludwig, P. Santen. **A Grid Service Discovery Matchmaker based on Ontology Description.** *Euroweb 2002 — The Web and the GRID: from e-science to e-business*, 2002

34.[Mahe00] M. Maheswaran and K. Krauter. **A Parameter-based approach to resource discovery in Grid computing systems**. *1st IEEE/ACM International Workshop on Grid Computing (Grid 2000)*, pp:181-190, 2000

35.[MicroUDDI] **Microsoft UDDI.** Retrieved December 1,2004, from http://uddi.microsoft.com/

36.[OGSA] **A visual tour of Open Grid Services Architecture.** Retrieved December 5, 2004, from http://www-106.ibm.com/developerworks/grid/library/gr-visual/index.html?ca=dgr-lnxw02TourOGSA

37.[Perfmon] **Perfmon.** Retrieved November 5,2004, from http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/nt_command_perfmon.mspx

38.[Rana01] O. Rana, D. Bunford-Jones, D. Walker, M. Addis,M. Surridge, K. Hawick. **Resource discovery for dynamic clusters in computational grids**. *IEEE Proc. of 15th International Parallel and Distributed Processing Symposium,* pp: 759 -767, 2001

39.[RDF] **RDF and Metadata.** Retrieved December 5, 2004, from http://www.xml.com/pub/a/2001/01/24/rdf.html

40.[RDF01] **Understanding RDF.** Retrieved November 5, 2004, from http://www.ilrt.bris.ac.uk/discovery/2001/01/understanding-rdf/

41.[RDF-FAQ] **Frequently Asked Questions about RDF.** Retrieved November 5,2004, from http://www.w3.org/RDF/FAQ

42.[Rosz98] M. Roszkowski and C. Lukas. **A Distributed Architecture for Resource Discovery Using Metadata.** *D-Lib Magazine: ISSN 1082-9873*, Internet Scout Project, Computer Sciences Department, University of Wisconsin-Madison, 1998

43.[RSS] **RSS.** Retrieved November 5, 2004, from http://blogs.law.harvard.edu/tech/rss

44.[SUNUDDI] **SUN Microsystems UDDI.** Retrieved November 5, 2004, from http://wwws.sun.com/software/xml/developers/uddi/

45.[UDDI] **UDDI.** Retrieved November 5, 2004, from http://www.uddi.org

46.[UDDI00] **Universal Description, Discovery and Integration (UDDI): Technical White Paper 2000**. Retrieved November 5, 2004, from http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

47.[Verb02] J. Verbeke, N. Nadgir, G. Ruetsch,I. Sharapov. **Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment.** *Proc of Third International Workshop on Grid Computing – GRID 2002*, Baltimore, MD, USA, , LNCS 2536, pp: 1 ff,Springer-Verlag, 2002

48.[WMI] **WMI Architecture.** Retrieved November 5, 2004, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/wmi_architecture.asp

49.[Wols99] R. Wolski, N. Spring, J. Hayes. **Predicting the CPU availability of time-shared Unix systems on the computational grid**. *Proc of IEEE Eighth International Symposium on High Performance Distributed Computing*, pp: 105 -112, 1999

50.[WQL] **Querying with WQL.** Retrieved December 5, 2004, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/wmi/querying_with_wql.asp

51.[WSA04] **Web Services Architecture**. *W3C Working Group Note 11 February 2004.* Retrieved November 5, 2004, from http://www.w3.org/TR/ws-arch/

Vita Auctoris

Name:	Mohammad Aktaruzzaman
Place of Birth:	Bhola, Bangladesh
Year of Birth:	1979
Education:	
	North South University Dhaka, Bangladesh 1997 – 1999
	University of Windsor Windsor , Ontario 2000 – 2001 B.Sc.
	University of Windsor Windsor , Ontario 2002 – 2004 M.Sc.