Non-Uniform Information Dissemination for Performance Discovery in Computational Grids

by

Dhawal B. Patel

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Technology

in

Information and Communication Technology

to

Dhirubhai Ambani Institute of Information and Communication Technology

May, 2005

DA-IICT

Declaration

This is to certify that

1. the thesis comprises my original work towards the degree of Master of Technology in Information and Communication Technology at DA-IICT and has not been submitted elsewhere for a degree,

2. due acknowledgement has been made in the text to all other material used.

Dhawal B. Patel

Certificate

This is to certify that the thesis work titled *Non-Uniform Information Dissemination for Performance Discovery in Computational Grids* has been carried out by *Dhawal B. Patel* (200311029) for the degree of Master of Technology in Information and Communication Technology at this Institute under my supervision.

Prof. Sanjay Chaudhary

Acknowledgements

I am grateful to numerous local and global “peers” who have contributed towards shaping this thesis.

At the outset, I would like to express my appreciation to Prof. Sanjay Chaudhary for his advice during my M. Tech research endeavor for the past one year. As my supervisor, he has constantly forced me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. I thank him for providing me with the opportunity to work with him. I am grateful to my thesis evaluation committee members (Prof. Naresh D. Jotwani and Prof. Kalpesh Kapoor) for providing useful suggestions for the research work.

I would like to express my sincere thanks to Mr. Klaus Krauter (Redfern Networks, Australia), Mr. Vishal Iyenger (M. Tech student, State University of New York, Binghamton),Mr. Sameer Tilak (Ph. D Candidate, State University of New York, Binghamton) for helping me to understand their own proposed models and providing me their valuable comments on the research.

I greatly appreciate Mrs. Padmaparna Haldar (VINT project,USC,CA) who greatly helped me to solve critical problems in Network Simulator ns-2 and GT-ITM topology generator. I acknowledge Dr. Damien Magoni (Assistant Professor, Université Louis Pasteur, LSIIT,France) for troubleshooting the problems in Network Manipulator NEM, a synthetic network topology generator.

I would like to thank DA-IICT for providing me the resources to pursue my research work.

I acknowledge my classmates for providing me the constructive criticisms through exhaustive discussions.

Last, but not least, I would like to dedicate this thesis to my family for their love, patience, and understanding — they allowed me to spend most of the time on this thesis.

Contents

Abstract

The required service in any resource-sharing environments like Grid, Peer-to-Peer etc., is discovery of resources. A resource discovery mechanism returns locations of resources that match the description, given a description of resource desired. Two resource-sharing environments are well defined with respect to target communities, resources, applications, scalability and fault tolerance: Grid and peer-to-peer systems. Grids are sharing environments that rely on persistent, standards-based service infrastructure that allow location independent access to resources and services, which are provided by geographically distributed machines and networks. The design of the resource discovery approach must follow the rules imposed by the characteristics of grid environment. These characteristics are 1. Independence from central global control, 2. Support for attribute-based search, 3. Scalability, 4. Support for intermittent resource participation. Depending upon the types of resources that are shared, the grids can also be of different types, e.g. computational grids for the environment in which only computational resources are shared, data grids for the one in which data are shared. The focus of thesis is on performance discovery in computational grids.

Grid schedulers, that manages the resources, requires up-to-date information about widely distributed resources in the Grid. This is a challenging problem given the scale of grid, and the continuous change in the state of resources. Several non-uniform information dissemination protocols have been proposed by researchers to efficiently propagate information to distributed repositories, without requiring flooding or centralized approaches. Recently, a new concept called the "Grid potential" proposed in [32], as the first step towards the design of non-uniform information dissemination protocols.

In this thesis, four non-uniform dissemination protocols are analyzed for computational grids based on the concept of "Grid potential" [32], which follows above-mentioned requirements for resource discovery. These protocols disseminate resource information with a resolution inversely proportional to the distance of resources from the application launch point. The performance evaluation is done with respect to the dissemination efficiency and message complexity. The results indicate that these protocols improve the performance of information dissemination compared to uniform dissemination to all repositories.

List of Abbreviations

API	Application Program Interface
AWARE	Adaptive Wide Area Resource Environment
CAN	Content Addressable Networks
CBR	Constant Bit Rate
DHT	Distributed Hash Table
DIS	Distributed Interactive Simulation
DUROC	Dynamically Updated Request Online Co-allocator
FLOPs	Floating point OPerations
GRIP	Grid Information Protocol
GRRP	Grid Registration Protocol
GSH	Grid Service Handle
GSR	Grid Service Reference
LDAP	LightWeight Directory Access Protocol
MDS	Meta Directory Services
MPI	Message Passing Interface
ND	Name-Dropper
OGSA	Open Grid Service Architecture
RTP	Real-Time Protocol
RTT	Round Trip Time
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
URL	Uniform Resource Locator
VINT	Virtual Internetwork Testbed
VO	Virtual Organization
WSDA	Web Service Discovery Architecture
WSDL	Web Service Description Language
XML	eXtensible Markup Language
PDP	Prioritized Dissemination Protocol

P2P	Peer to Peer
RD	Randomized Protocol
SWP	Swamping algorithm
NH	Neighborhood algorithm

List of Figures

Chapter 1

Introduction

The advantage of the Internet is the opportunity to share, not only the communication infrastructure but also other resources, such as computers, softwares, scientific instruments, sensors, data etc. The classic example is Seti@home: a project from Berkeley that benefits from computational power of computers on the Island of Man, Tonga, Saint Pierre, and Miquelon. Another example is the peer-to-peer, or P2P, music sharing networks such as Napster, Kazaa, or Gnutella. 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 behavior, and the inherent lack of reliability.

Sharing resources over the Internet also has many names and objectives: the Web is a form of resource sharing, but so are Grid, global computing, Peer-to-Peer (P2P) computing, or utility computing, to name just a few.

Two resource-sharing environments have particularly well defined applications and communities: Grids and P2P systems. Grids provide software and hardware infrastructure to integrate resources from multiple administrative domains in support of a variety of data and compute intensive applications.

P2P systems are Internet applications that integrates the resources of a large number of autonomous participants. They have emerged as vertically integrated popular applications such as music sharing and grew tremendously in user-base over a short time. The scale of current P2P systems make them an interesting case study: a community with millions of users giving significant insight into user behavior, heterogeneity, and reliability. Thus, the scalability feature of P2P systems makes it a perfect candidate to be utilized in designing the Grid systems.

In all above systems, one important functionality is resource discovery: given a resource description, the resource discovery component returns the location of one or more resources that fit the description. However, the resource discovery problem has slightly different meaning and requirements depending on the particular resource-sharing environment and therefore, its objectives and constraints. The focus of this thesis is on performance discovery in computational grid environment.

The large number of resources coupled with heterogeneity can help programmers match their applications to the resources that are best suited to run them. The suitability could be determined based following factors:

1. Memory requirements
2. Processor speed requirements
3. Proximity to necessary data sets and other suitable computational nodes

Importantly, this list can be affected by dynamic properties of the computational nodes. Current load averages affect CPU performance; the amount of available memory varies as processes start up, claim memory, free it, and terminate; data sets move, and processes migrate. To make effective application placement decisions, and indeed to realize the potential of computational grids in solving larger problems more efficiently, grid schedulers. The grid schedulers are used to map application objects to computational resources. The grid schedulers must have access to up-to-date information about grid resources.

The basic approaches to collecting and discovering resource information that may be appropriate for limited size single-site distributed systems will not scale with the expected number of resources, applications, and resource discovery queries in grids. In particular, direct querying remote resources would require too many messages to remote locations, and does not address the problem of discovering the names of the resources to query. Alternatively, centralized information repositories that are pro-actively updated with grid resources information would attract too much traffic, due to both updating of this information and the queries against it.

To fulfill the above requirements, the resource discovery service uses status databases that are maintained by network-wide information services. For scalable implementations, it is essential to organize the status databases in a distributed fashion. With a distributed organization for the status databases, the queries can be executed very efficiently, but the updates to the databases may be costly. The updates to the status databases could become costly in terms of message overhead and communication time in keeping the status database consistent. Most of the update costs are caused by the communication operations performed to disseminate status information across the Grid. This thesis focuses on approaches for performing the data dissemination that is necessary to keep the status databases consistent.

Recently, data dissemination for resource discovery has been an active research area with applications such as web content management motivating the research. Several research initiatives have examined this issue in the context of Grid computing as well. Resource discovery process is mainly used by a node to find out the best set of candidate computational resources that can execute a job or provide a specific service. This problem can be considered as a special case of the resource discovery and it can be called as “Performance Discovery.” The efficiency of the performance discovery process can be improved by observing that a Grid is likely to consist of a variety of heterogeneous machines and networks and some of the machines will be more significant than others in terms of performance. Therefore, one way of reducing the overhead involved with data dissemination would be to control the extent of status dissemination of a machine according to the significance of the machines. The concept of “Grid potential” introduced in [32] , encapsulates the relative processing capabilities of the different machines and networks that constitute the Grid. The analysis of the protocols in this thesis is based upon this concept.

1.1 Grid Computing

As computation, storage, and communication technologies steadily improve, increasingly large, complex, and resource-intensive applications are being developed both in research institutions and in industry. It is a common observation that computational resources are failing to meet the demand of such applications. The power of network, storage, and computing resources is projected to double every 9, 12, and 18 months, respectively. As noted in [17], these three constants have important implications. Anticipating the trends in storage capacities (and price), application developers and users are planning increasingly large runs that will operate on and generate petabytes of data. Although microprocessors are reaching impressive speeds, in the long run they are falling behind the storage. As a result, it is becoming increasingly difficult to gather enough computational resources for running applications at a single location. Fortunately, improvements in wide-area networking make it possible to aggregate distributed resources in various collaborating institutions and to form what have come to be known as Computational Grids (or Grids). To date, most Grid applications have been in the area of scientific computing as scientists worldwide are resorting to numerical simulations and data analysis techniques to investigate increasingly large and complex problems.

The term Grid was coined in the late 90s [18] to describe a set of resources distributed over wide-area networks that can support large-scale distributed applications. The analogy likens the Grid to the electrical power grid: access to computation and data should be as easy, pervasive, and standard as plugging in an appliance into an outlet.

1.1.1 What is a Grid?

In the foundational paper "The Anatomy of the Grid" [19], Foster, Kesselman, and Tuecke attempt to address the problem discussed in previous subsection by redefining the Grid problem as "*coordinated resource sharing and problem solving in dynamic, multi-institutional, virtual organizations*." The word "resource" is to be taken in a broad sense, to include data, computers, scientific instruments, softwares, sensors etc. The concept of a virtual organization (VO) is central to Grid computing. The VO is explained in detail in the following subsection. Further, the grid system must.

1. Coordinate resources that are not subject to centralized control. Sharing must be "coordinated" in which resources together with their providers/consumers are clearly defined, and in that multiple resources may need to be organized in an integrated fashion to achieve qualities of service. Achieving this coordination involves the establishment and enforcement of sharing agreements.

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

3. 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 response time, throughput, availability and security, and/or co-allocation of multiple resource types to meet complex user demands.

1.1.2 Virtual Organizations

A Virtual Organization (VO) is a set of participants with various relationships that wish to share resources to perform some task. In [19], Foster et al. argue that the Grid problem is thus central not only to "e-science," but also to industry, where the coordination of distributed resources both within and across organizations is increasingly important. [19] identifies and defines a set of common requirements for enabling VOs. One critical observation is that a simple client-server model is not suitable for enabling most VOs. Instead a spectrum of architectures ranging from client-server to general peer-to-peer is necessary as participants are alternatively resource providers or consumers. Another observation is that currently available distributed computing technologies are not appropriate to enable VOs. Either these technologies do not support the wide variety of required services and resources, or they suffer from lack of flexibility and control needed for enabling the type of resource sharing necessary. As a result, there is a need for defining a new technology to support VOs: a Grid software infrastructure. This is discussed in the following subsection.

1.1.3 Grid Architecture

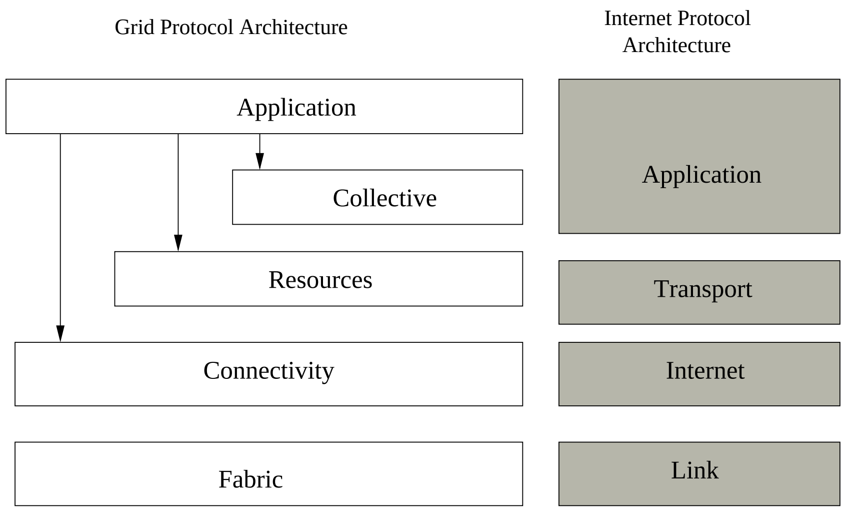

Figure 1.1: The layered Grid architecture and its relationship to the Internet protocol architecture reproduced from [19]

The software infrastructure presented in [19] places a large emphasis on interoperability, as it is fundamental to ensure that VO participants can share resources dynamically and across different platforms, programming environments, and languages. To achieve interoperability, it is necessary to specify Grid protocols. These protocols can then be implemented as a part of Application Programming Interfaces (APIs) and Software Development Kits (SDKs) to provide layered programming abstractions. We review Grid architecture layers in detail in this subsection. The Grid fabric provides the lowest level of access to actual resources (e.g. computer, disk, file system, cluster of computers) and implements the mechanisms that allow those resources to be utilized. More specifically, those mechanisms must at least include state enquiry and resource management mechanisms, each of which must be implemented for a large number of native systems. The Grid connectivity layer defines communication, security, and authentication protocols required for network transactions between resources. The Grid resource layer builds on the connectivity layer to implement protocols that enable the use and sharing of individual resources. More specifically, two fundamental components are

1. Information protocols for querying the state of a resource and
2. Management protocols to negotiate access to a resource.

While the Resource layer is focused on interactions with a single resource, the next layer in the architecture contains protocols and services that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. The services like Co-allocation, scheduling, and brokering services, Monitoring and diagnostics services for VO, Data replication services, Workload Management systems and collaboration frameworks are included in the Collective layer. Finally, the application layer is where VO applications are implemented and may use several of the previous layers. The resulting architecture is depicted in Figure 1.1. The description of this architecture in [19] uses components of the Globus toolkit [2] as a concrete example of Grid layer implementations. Globus is a large open-source, community-based effort at the Argonne National Laboratory and the Information Science Institute; Globus provide a software infrastructure that is currently leveraged by most Grid efforts. Beyond the Globus toolkit, the Global Grid Forum [1] is a consortium of over 1000 academic researchers and industrial partners whose goal is to make recommendations for Grid infrastructure development.

From the above architecture, the requirements for the grid infrastructure can be figured out as follows

1. Resource enquiry and discovery
2. Resource Management
3. Grid Security

These requirements must be fulfilled under the following constraints

1. Number of resources
2. Heterogeneity of resources
3. Heterogeneity in platform architecture
4. Site autonomy (Resources owned by different organizations in different administrative domains and local policies for usage, scheduling and security)
5. Network load and bandwidth
6. Interconnect Network (For loosely coupled systems hardly ever matters since it is the internet)

This makes the design of grid infrastructure complex.

1.1.4 Grid Applications

Three major application classes for Grids are described briefly in this subsection.

Distributed Supercomputing: Distributed supercomputing applications use grids to aggregate substantial computational resources in order to tackle problems that cannot be solved on a single system. Some of the contemporary examples: Distributed interactive simulation (DIS), simulation of complex physical processes.

High-Throughput Computing: In high-throughput computing, the grid is used to schedule large numbers of loosely coupled or independent tasks, with the goal of putting unused processor cycles to work. Examples are Platform computing, Condor system.

On-Demand Computing: On-demand applications use grid capabilities to meet short-term requirements for resources that cannot be cost-effectively or conveniently located locally. These resources may be computation, software, data repositories, specialized sensors and so on. For example: The NEOS [26] and Netsolve [37]

Data Intensive Computing: In data-intensive applications, the focus is on synthesizing new information from data that is maintained in geographically distributed repositories, digital libraries, and databases.

1.1.5 Grid Communities

Who will use grids? One approach to understanding computational grids is to consider the communities that they serve.

Government: This comprises of thousands or perhaps tens of thousands – of officials, planners, and scientists concerned with problems traditionally assigned to national government, such as disaster response, national defense, and long-term research and planning.

A Health Maintenance Organization: In this example, the community supported by the grid comprises administrators and medical personnel located at a small number of hospitals within metropolitan area. The resources to be shared are a small number of high-end computers, hundreds of workstations, administrative databases, medical image archives, and specialized instruments such as MRI machines, CAT scanners etc.

A Materials Science Collaboratory: The community includes a group of scientists who operate and use a variety of instruments, such as electron microscopes, particle accelerators, and X-ray sources, for the characterization of materials.

Computational Market Economy: This community comprises the participants in a broad-based market economy for computational services. It includes the providers of compute resources; providers of specialized services, such as financial modeling, graphics rendering, and interactive gaming; network providers, who contract to provide certain levels of network service; and various other entities such as banks and licensing organizations.

1.2 Resource Discovery in Grid

When dealing with large sets of entities, the basic problem is locating one or more entities that match a given description. These descriptions can have various forms, from globally unique identifiers (such as a person's name and address) to enumerations of desired attributes, such as the required skills of a qualified candidate presented in job advertisements. As already suggested, everyday life is filled with instances of resource discovery: a lookup in the Yellow Pages; a Google search; posting an advertisement in a newspaper; browsing the Web by specifying the desired URL. Given the characteristics of each instance, different solutions are employed: centralized indexes (as in telephone books or Google), targeted broadcast (as in newspaper advertisements), or mechanisms that associate human-readable addresses with machine-understandable locations (as in DNS). We study the resource discovery problem in a resource-sharing environment that preserves the essence of Grids but has the scale and dynamism of peer-to-peer communities

1.2.1 Resource Discovery and Graph theory

Let E be the edge set of Graph G and elements of this set are called edges. According to [30], a network can be viewed as a directed graph and if u, v are any two nodes of a network and if $u\ knows\ v$ then an edge between u and v is created, $(u, v)\ \epsilon\ E$. For any node u, if $\Gamma(u)$ is the known set of u then (u,v) $\epsilon\ E\ iff\ v\ \epsilon\ \Gamma(u)$. The goal of resource discovery algorithm based on a graph theory is to develop a complete knowledge graph or strongly connected graph from a given weakly connected graph. A weakly connected graph is a directed graph in which if we ignore the edge directions then the graph is connected. A strongly connected graph is a directed graph that has a path from each vertex to every other vertex.

1.2.2 Requirements for Resource Discovery

As we have noted, we expect Grid and P2P systems to converge in a unified resource sharing environment. This 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 patterns will be highly variable: there will be perhaps a larger number of stable nodes than in today's P2P systems, but many resources will join and leave the network frequently. Resources will have highly diverse types (computers, data, services, instruments, storage space) and characteristics (e.g., operating systems, number of CPUs and speed, data of various sizes, services). Some resources will be shared following well-defined public policies, such as "available to all from 6 pm to 6 am." Other resources will participate rather chaotically, for example, "when idle." Technical support will be variable: some participants will benefit from technical support, whereas others will rely on basic tools provided by the community (e.g., today's Gnutella nodes run various implementations of the protocol, each with its own peculiarities). In addition to the requirements imposed by the resource-sharing environment, various challenges are raised by usage scenarios. Resource discovery may be expected to provide browsing capabilities: users may want to learn what resources are shared in a virtual organization, may be to decide whether to join it with their own resources. Also, resource discovery may be employed to support other mechanisms, such as scheduling or monitoring services: when jobs are submitted, a scheduler may call resource discovery to get up-to-date information about available, appropriate resources. A monitoring service may want to detect all existing replicas of a file in order to decide whether more replicas are needed for improving performance or availability. Another challenge comes from the diversity of requests, from simple requests that specify one resource, such as a file named "da-iict" to aggregated requests, such as 100 Linux machines separated by at least 20ms. latency pair wise. However, the design of a mechanism that meets the desired usage scenarios must obey the rules imposed by the characteristics of the resource-sharing environment. These characteristics require a resource discovery mechanism with the following features:

1. Independence from central, global control. This is a departure from previous Grid solutions and a step toward the fully decentralized solutions typical of P2P approaches.

2. Support for attribute-based search, a feature not found in current P2P solutions. That is, solutions must support requests that specify a set of desired attributes (Linux machine with more than 128 MB of available memory) rather than be designed around globally unique identifiers. Attribute-based search is even more challenging when resource attributes can vary over time (e.g., CPU load, available bandwidth, even

software versions) and when the number of resources is large and/or dynamic (as resources join, leave, or fail).

3. Scalability, which becomes more important with increased scale and participation. Scalability refers not only to the number of hosts in network, but also to the number of messages needed to discover a resource

4. Support for intermittent resource participation, a characteristic frequent in today's P2P systems but rare in current Grid solutions.

1.2.3 Components of Resource Discovery

[25] identified four different architectural components for a general resource discovery solution, "Membership protocol," "Overlay construction," "Preprocessing," "Request processing." Iamnitchi 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."

1.2.3.1 Architectural Components

According to [25], following architectural components are identified:

Membership Protocol: Specifies how a new node becomes a member of an unstructured network and how a member can learn about other members in such network. It is responsible for collecting and updating information about currently active members.

Overlay Construction Function: It 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.

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

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. The data dissemination for resource discovery task comes as a part of "Preprocessing" stage as per described above.

1.3 Problem statement

This thesis analyzes the approaches for scalable grid resource discovery using replicated information repositories that are updated non-uniformly. The presented ideas are based on the concept of “Grid potential” proposed in [32]. In particular, non-uniform dissemination (as opposed to full dissemination) of resource information is used to reduce the overhead of uniform information replication while maintaining accurate information at locations where it is most likely to be needed. The observation, that grid resources tend to be of more interest to nearby users, because the overhead in starting the job and transferring data and results increases with the distance to the resources, is leveraged. The dissemination protocols work by propagating the resource state information more aggressively and in more detail to nearer information repositories than they do to farther ones. Thus, repositories have more accurate and fresher information regarding nearby resources with less accurate and fresh information about distant resources.

The simulation study has been performed to evaluate the performance of these protocols.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2 and 3 reviews the current developments for resource discovery in large distributed networks and Grid computing in specific.

Chapter 4 explains the simulation setup for the measuring the performance of presented algorithms in network simulator ns-2 and also presents the details of extensions made to the network simulator ns-2.

Chapter 5 presents the test cases for simulation, simulation results and analysis of results for all algorithms under analysis.

Chapter 6 presents the conclusions and provides directions for future work.

Chapter 2

Literature Survey

The resource discovery problem comprises several necessary components. Iamnitchi and Foster in [25] have defined four axes of the solution space, which is enlisted in section 1.2.3 of this thesis. Relative to Iamnitchi and Fosters' taxonomy, the work in this thesis is best viewed as preprocessing that is intended to make query processing more efficient. The information dissemination protocols presented in this thesis can be viewed as orthogonal to the overlay topology of forwarding nodes. However, the overlay topology of forwarding nodes could have significant influence on the effectiveness of the protocols. The final deployed solution must make the forwarding probabilities sensitive to the overlay topology. In this chapter, the detailed discussion of available algorithms, protocols and approaches for resource discovery in distributed networks in general is presented. Also, the applicability of each of the above is tested for suitability in Grid resource discovery.

2.1 Algorithms

As discussed in section 1.2.1, the relation between resource discovery and graph theory that an instance of the resource discovery problem is modeled as a directed graph. Each machine is represented by a node of graph and edges represent the "knowledge" relation which is defined as, if $(x, y) \epsilon k$, where k is "knowledge" relation, then x knows y, where x and y are nodes. Also, it is noted that "knowledge" relation is *reflexive, not symmetric* and *not transitive* in nature, i.e. it is *reflexive only* relation. Edges are added to the graph when a machine knows each other. Harchol et al. in [22] have proposed several randomized algorithms for resource discovery in distributed networks in general. These algorithms are discussed briefly in this section. The assumption taken in these algorithms is that the "knowledge" graph is initially weakly connected and "physical" network graph is strongly connected. The performance of these algorithms is measured in terms of pointer communication complexity, connection communication complexity and number of rounds. The parameters defined in [22] are as follows:

Pointer communication complexity is defined to be the total number of pointers communicated during the course of algorithm.

Connection communication complexity is defined to be the total number of connections, which are opened during the course of algorithm, where a connection between u and v is created when u contacts v, u and v are nodes.

Number of Rounds one round is defined as the time for each machine in the network to contact one or more of its "logical" neighbors and exchange some subset of its neighbor list.

The running time of a resource discovery algorithm is the number of rounds required until every machine knows about every other machine, i.e. a complete knowledge graph is formed. Different rounds can have different running times. One difference is caused by the fact that the "neighbors" of a node may be of different distances from the node in the underlying physical network. However, it can also turn out that communicating between two nodes which are physically close may be more costly than communicating between two nodes which are far apart because of differences in the speeds of the routers and speeds of the machines. Thus, simply the number of rounds is used as a run time metric.

2.1.1 Flooding Algorithm

In Flooding algorithm [22], a node is initially configured to have a fixed set of neighboring nodes. The direct communication is only allowed with nodes in this set and new edges that are added to the graph are not used for communication. [22] analyzed the flooding algorithm. Let $\Gamma(v)$ the set consisting of v and of all the nodes that v points to. In every round of the Flooding algorithm, each node v contacts all of its initial neighbors and transmits to them the updates to $\Gamma(v)$ (denoted by $\Gamma(v)^{updates}$). A node u that receives the updates then updates its set of neighbors by merging $\Gamma(u)$ and $\Gamma(v)^{updates}$.

2.1.2 The Swamping Algorithm

According to [22], the swamping algorithm is identical to the Flooding algorithm except that machines may now open connections with all their current neighbors, not just their initial neighbors. Since the neighbor sets change, all of the current neighbor set is transferred, not just the updates.

2.1.3 The Random Pointer Jump Algorithm

In this algorithm, in each round, each node contacts with a random neighbor, and then this random neighbor sends all of its neighbors to the sender node. Finally, sender neighbor and random neighbor's neighbors get merged. Harchol et al. in [22] claimed that a strongly connected graph with n nodes needs $O(n)$ complexity time to converge to a complete graph.

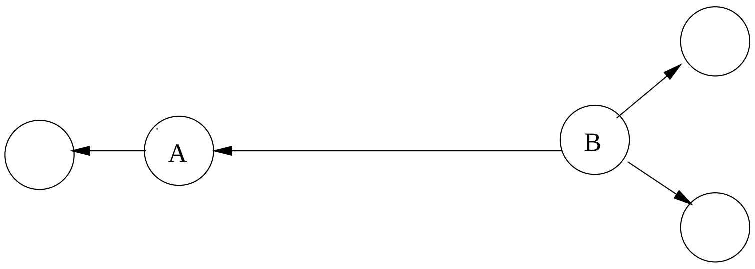

Figure 2.1: Before random pointer jump

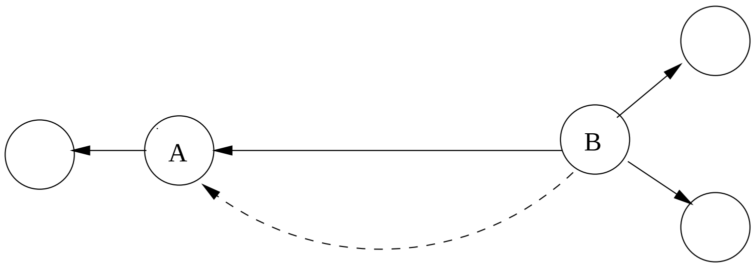

Figure 2.2: After random pointer jump

As shown in Figure 2.1 before the random pointer jump, the node B chooses at random one of its neighbors and opens a connection with it. Here A is the chosen neighbor. In 2.2, after the Random Pointer Jump node A has passed to node B, all of its neighbors. Now B also points to the neighbor of A. The dashed lines indicate newly formed edges.

2.1.4 The Name-Dropper algorithm

Harchol et al. in [22] proposed "Name-Dropper" algorithm for resource discovery, which takes $O(log^2n)$ rounds to learn each other, $O(n^2log^2n)$ number of connections & $O(n^2log^2n)$ number of pointers in a network. In [22] the authors claimed that this new algorithm is more efficient in terms of required time and total number of network communications compared to three other algorithms, mainly flooding algorithm, swamping algorithm, random pointer jump algorithm. Harchol et al. also claimed that this algorithm achieves "near-optimal performance both with respect to time complexity and with respect to the network communication complexity." Due to this characteristic of Name-dropper, this thesis also focuses on the study of the similar algorithm i.e. randomized protocol for information dissemination in grid environment specified by [32]. The Name-Dropper algorithm is almost identical to "Random pointer" algorithm. But according to Harchol et al. in worst-case $O(log^2 n)$ rounds of "Name-Dropper" algorithm dominates over $O(n)$ rounds of complexity time of "Random pointer" algorithm, where n is the number of machines. As shown in Figure 2.3

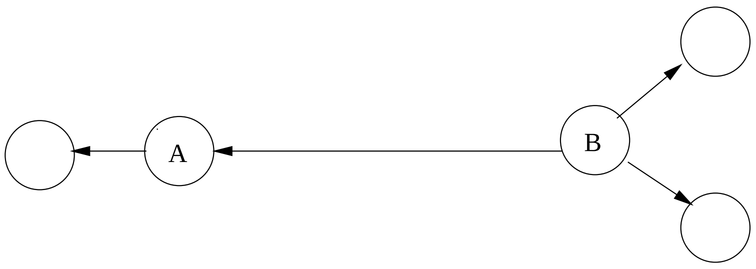

Figure 2.3: Before Name Dropper

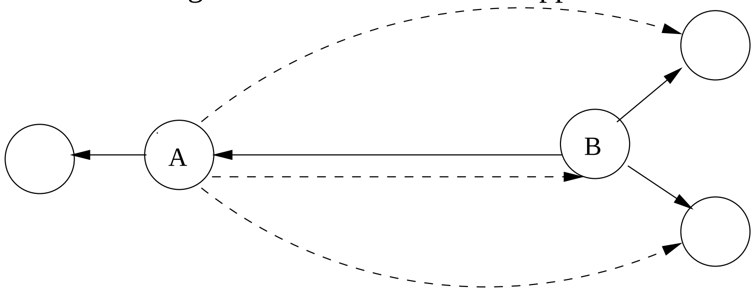

Figure 2.4: After Name Dropper

before Name-Dropper node B chooses a random neighbor; here A is the chosen neighbor. In Figure 2.4 after one round of the Name-Dropper algorithm B has passed to A all of its neighbors and A has added edges to these neighbors. In addition, A learns about B. The dashed lines indicate newly formed edges.

2.1.5 Comparison of algorithms

The Table 2.1.5 contains the performance comparison of all algorithms.

n = number of vertices/nodes in the graph
$d_{initial}$ = initial diameter of the graph.
$m_{initial}$ = the number of edges in the initial graph.

Algorithm	Num. Rounds	Pointer Communication	Connection Communication
Flooding	$d_{initial}$	$\Omega(n.m_{initial})$	$\Omega(d_{initial}.m_{initial})$
Swamping	$O(logn)$	$\Omega(n^3)$	$\Omega(n^2)$
Random Pointer Jump	$\Omega(n) in worst case$	$(num.rounds).\ m_{initial}$	$(num.rounds)$.n
Name-Dropper	$O(log^2 n)$	$O(n^2 log^2 n)$	$O(nlog^2 n)$

Table 2.1: Performance comparison of four algorithms. Reproduced from [22]

2.2 Protocols

Similar to the Routing protocol, which is defined as “A series of periodic or on-demand messages containing routing information that is exchanged between routers to exchange routing information and provide fault tolerance,” the protocols for resource discovery at application layer have also been designed. In this section, the review of current protocols for resource discovery in grid is presented.

These protocols accomplish the resource discovery through the implementation of a specific resource discovery algorithm.

2.2.1 GRid Information Protocol

According to [11], GRid Information Protocol (GRIP) is used to retrieve information of entities in VO. According to [11], this protocol is one of the main fundamental building blocks of VO. A user or more frequently an aggregate directory or other program, uses GRIP to obtain information from an information provider about the entity(s) on which the provider possesses information. GRIP supports both discovery and enquiry. As shown in Figure 2.5 Discovery is supported with the help of a search capability. Enquiry corresponds to lookup of information. The LDAP is adopted as the protocol for GRIP. LDAP defines a data model, query language, and wire protocol.

The basic limitation of this protocol is that no specification about the algorithm to be used for resource discovery is specified.

Using the GRid Information protocol (GRIP), users can query aggregate directory services to discover relevant entities, and/or query information providers to obtain information about the individual entities.

Description services are normally hosted by a Grid entity directly, or by a front-end gateway serving the entity.

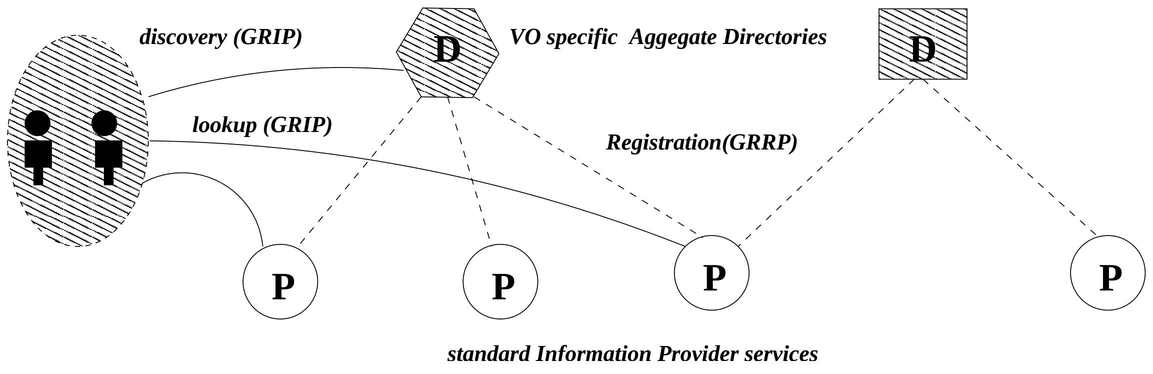
discovery (GRIP)
D
VO specific Aggegate Directories
D
lookup (GRIP)
Registration(GRRP)
P
P
P
P
standard Information Provider services

Figure 2.5: MDS-2.0 Architecture overview. Reproduced from [11]

2.2.2 Grid Registration Protocol

According to [11], Grid Registration Protocol (GRRP) is responsible for notifying aggregate directory services which refer to VO-specific resources, which are found by GRIP protocol. This is shown in Figure 2.5. Both GRIP and GRRP are used by Globus metacomputing toolkit [16]. In [5], they also use GRIP protocol of MDS to discover the resource in proposed adaptive resource selection mechanism. The authors of [5] claimed 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 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 [15], a LDAP server speaks LDAP protocol and it also makes response to the query. [11] showed how Lightweight Directory Access Protocol (LDAP) enabled Globus Monitoring and Discovery Service (MDS) can be used as an information service, which is responsible for providing current availability and capability of resources.

2.2.4 Simple Object Access Protocol

According to [15], SOAP is an enveloping mechanism for carrying XML data. 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

2.3.1 Peer-to-peer Approach

“Peer-to-peer” refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner. P2P networks can be classified based on the control over data location and network topology. There are three categories: unstructured, loosely structured, and highly structured. In an unstructured P2P network such as Gunutella [40], no rule exists which defines where data is stored and the network topology is arbitrary. In a loosely structured networks such as FreeNet [9], the overlay structure and the data location are not precisely determined. In a highly structured P2P network such as CHORD [13], CAN [39], PASTRY [41], both the network architecture and the data placement are precisely specified. The neighbors of a node are well defined and the data is also stored in a well-defined location. The highly structured network requires much less message complexity then the unstructured networks still provide the fault tolerance and scalability compared to loosely structured networks. The highly structured network uses DHTs and Consistent hashing [29] to implement the lookup. The Chord systems takes only $O(logN)$ messages to be exchanged with high probability, where N is the number of servers in the chord network. We test the applicability of this system in resource discovery in Grids.

The consistent hashing used in above systems, assigns each node and key an m-bit identifier. A node’s identifier is defined by hashing the node’s IP address. Hashing the key produces a key identifier
$ID(node) = hash(IP, Port)$
$ID(key) = hash(key)$.

The data key can be taken as attribute and attribute values of the resource in grid. Thus, the attribute values can be mapped to the data key and can be applied to [13], but this type of mapping is highly sensitive to the dynamic attribute values such as those describing the compute resources. It would limit the type of queries that could be performed. Here, both the domain and range of the consistent hash function are dynamic. Thus, utilizing the strategies listed above will not be suitable for this application. With this justification, the unstructured peer-to-peer networks are used to design the protocols in this thesis. [25] discussed peer-to-peer resource discovery in detail. [25] proposed peer-to-peer resource discovery architecture for a large collection of resources. The authors of [25] claimed 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. In [25], author discussed different resource discovery problems in a large distributed resource-sharing environment especially in a grid environment. In this document, author identified four different architectural components called "Membership protocol," "Overlay construction," "Preprocessing," 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." In [25], the authors gave a brief description of different resource discovery approaches in peer-to-peer networking. The authors claimed that using four axes framework [25], it is possible to design any resource discovery architecture in a grid.

2.3.2 Decentralized Approach

Iamnitchi in [25] and Rana in [38] discussed this approach. [38] described a decentralized resource management and discovery architecture based on interacting software agents where an agent can represent a service, an application, a resource or a matchmaking service. The authors of [38] showed that this proposed approach could provide dynamic registration of resources and user task. According to [38] paper, this approach is basically a matchmaking approach, which can facilitate dynamic resource management and resource discovery in a grid environment, based on XML documents that provides the resource availability and resource capability.

2.3.3 Request Forwarding Approach

According to [25], following four-request forwarding approaches are identified.

2.3.3.1 Random Walk Approach

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

2.3.3.2 Learning-Based Approach

As discussed in [25], a request is forwarded to a node who answered similar request before. If no similar answer is found, the request is forwarded to a randomly chosen node.

2.3.3.3 Best-Neighbor Approach

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

2.3.3.4 Learning-Based + Best-Neighbor Approach

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

According to [25], 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-neighbor 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 Discovery mechanisms on different architectures

2.4.1 Globus

Foster et al. in [15] presented 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 [15], the authors described data communication, resource discovery, resource allocation, and authentication. 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 behavior of metacomputing environments. Globus uses GRRP and GRIP protocols to discover the resources in VO.

2.4.2 Open Grid Service Architecture

Foster et al. in [16] described a web service enabled grid architecture called Open Grid Service Architecture (OGSA), which supports creation, termination, job management and invocation of unpredictable services through standard interfaces and conventions. The authors investigated three main web service standards mainly Simple Object Access protocol (SOAP), Web Services Description Language (WSDL), WS-Inspection. Because of dynamic nature of WSDL specially on "registering and discovering interface definition, endpoint implementation description" and generating proxy dynamically the authors chose WSDL as the main web service standard of OGSA. In OGSA, everything is referred to as a grid service and this grid service is expressed using Web Services Description Language (WSDL) for lifetime management, discovery mechanism and notification, etc. In OGSA, information about a Grid service which is referred as "service data" and facilitates helps "FindServiceData" to discover a resource. In [16] the authors represented OGSA as a natural evolution of Globus toolkit specially version 2. [16] also identified the solution to the question "how grid functionality can be incorporated into a Web services framework" which was raised in [15].

2.4.3 Web Service Discovery Architecture

Hoschek in [23] 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. WSDA and OGSA [16] 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.

2.4.4 Nimrod/G

Buyya et al. in [7] proposed a "component based architectural design for Nimrod-G" which was 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. The authors of [7] claimed that Nimrod/G could make good scheduling decisions.

2.4.5 Universal Description, Discovery and Integration

According to [43], 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 [43], 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.

2.5 Research Challenge

Grid Information service by its nature maintains highly dynamic data. The client demands the fresh data and fast query response time. Also, the lookup service must be scalable and fault tolerant. Since, the grid is developed on overlay peer-to-peer network, which is unstructured. In the direction to fulfill these requirements under the above mentioned constraints, this thesis uses the concept of "Grid potential" described in [32]. The details of "Grid potential" are discussed in chapter 3.

Chapter 3

Parameter Based Performance Discovery in Grids

[32] introduced the concept of “Grid potential.” “Grid potential” encapsulates the relative processing capabilities of the different machines and networks that constitute the Grid. The extent of dissemination in a Grid is adaptively controlled using this concept.

This thesis evaluates the four protocols based on “Grid potential.” The evaluation parameters are discussed in detail in chapter 4.

In this chapter, the four algorithms proposed by [32], are explained in detail.

The “Non-uniform” dissemination is the better approach for grid environments as, the grid is considered to consist unique or scarce resources, e.g. scientific instruments, supercomputers, cluster of CPUs etc. These resources are unique and scarce in Grid. For the grid application schedulers, to discover these unique resources, the information of these resources must be propagated widely compared to those resources, which are common in the Grid. This, attracts more traffic (resource requests) to unique or scarce resource and thus the unique requirements of most of the applications are satisfied and the resource will be used efficiently.

Propagating the resource information of unique or powerful resources more aggressively and widely, in turn reducing the propagation of relatively common resources, will save the message overhead. This increases the scalability of the system, maintaining comparable amount of accuracy in application scheduling decisions with that of uniform dissemination protocols.

The protocols analyzed in this thesis, filters the information at forwarding nodes/router-allocators and/or at endsystems based on one or more of several criteria.

Different protocols can be explored that filter data according to different criteria, e.g. probabilistically

- if data has not changed recently
- if data is changing too rapidly
- if data describes a common resource
- if the resource is too far from the source of request

Clearly wide range of protocols can be designed using the above ideas as the basis.

This research work investigates the performance of uniform dissemination algorithm, i.e. flooding algorithm and three non-uniform dissemination algorithms described in [32], in which one of them is a probabilistic protocol. The observed error, in the monitored information at the remote nodes relative to the actual value of the information at its source, is measured. Error graphs with characterization of message overhead – the amount of data that is propagated through the grid to achieve the reported error rates are coupled.

The experiments demonstrate that a large saving in overhead in information dissemination is possible without losing much accuracy. As a result, it is believed that non-uniform dissemination holds the promise of improving the scalability of grid resource discovery.

3.1 Grid Potential

In this section, "Grid potential" concept described in [32] is explained. The "Grid potential" is the basis for the dissemination protocols design.

Grid potential at a point in a Grid as described in detail in [32], captures the relative processing power (operations/sec) delivered to an application at that point in the Grid. The Grid potential depends on the machines that are present in the neighborhood and the networks used to connect them. This potential, however will decay as the launch point of the application moves away from the point at which machine is connected to Grid. The decay rate depends on total latency of network links used to connect them.

Grid potential is similar to TTL concept used in Internet. TTL could also be used to control the extent of dissemination of resource information but has following problems:

- TTL is fixed at the endsystems and thus, there is no involvement of intermediate nodes or router-allocaters
- The decay in TTL is not the function of network link capacities, which will not allow the grid application schedulers to reduce the response time of the application.

The above problems are significant as, it is considered that the application response time to be also one of the important parameter affecting the grid application performance. The grid application scheduler must select the resource for launching the application such that minimum response time is achieved, if the required capacity is discovered. To justify the point, consider an example shown in Figure 3.1 below. Let node 1,2,3, and 4 be the endsystem as well as router-allocaters in a computational grid connected with each other in the structure shown in Figure 3.1. The compute potential (L_i) of node i is shown with each node. Also, the compute potential of remote nodes seen by local node is shown by R_{ij} where, R_{ij} is the compute potential of j^{th} node seen by i^{th} node. The network links are annoted with propagation delay(ms) and bandwidth (Mbps).

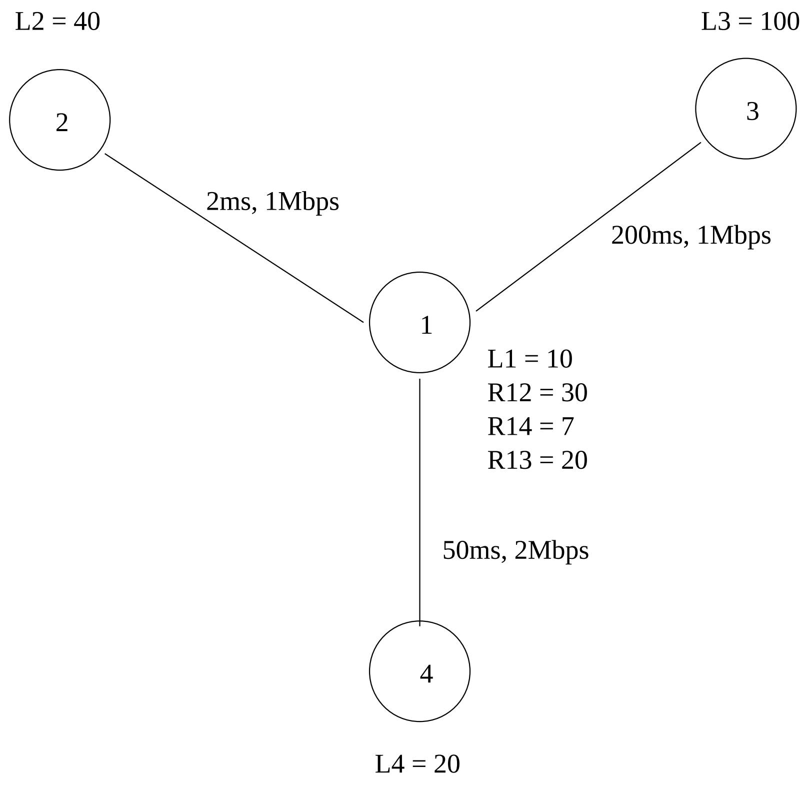

Figure 3.1: Grid potential decay affecting the response time of the application in a sample computational grid

It is assumed that node 1 knows the compute potential of node 2,3, and 4. The compute potentials of nodes 2,3, and 4 seen by node 1 is shown in Figure 3.1 by R_{12}, R_{13} and R_{14} respectively. The decay in the compute potentials of nodes 2,3 and 4 respectively is assumed to be the function of network link capacities(Total latency). The actual decay or potential drop will be discussed later in this chapter. If the resource request for launching an application comes at node 1, then the grid scheduler will decide to schedule the application on node 2 since, the capacity is available and the response time will be minimum as the total latency of the link between node 1 and node 2 is less compared to link between node 1 and node 3. The potential of node 3 is decayed from 100 to 20 (high rate), since the propagation delay of the link between node 1 and node 3 is very high (200ms) compared to link between node 1 and node 2 i.e. 2ms. Node 3 do also have the capacity to run the application, but the application response time will increase tremendously due to the total latency of the link between node 1 and node 3. Thus, the decay of compute potential is taken to be the function of network link capacities to reduce the application response time.

The calculation or determination of the grid potential is based on rate-based attributes like CPU speed, FLOPs etc. of the node. The rate-based attributes are calculated by the benchmarking programs for each attribute. The mathematical expressions to calculate the compute potential of node based on these rate-based attributes, as described in [32] is given in Appendix A.

3.1.1 Node component potential

Node component potential with respect to a rate-based attribute is the number of operations performed by a node in a second measured by corresponding benchmark program. The mathematical expressions are given in Appendix A.

3.1.2 Weighted Node component potential

The Weighted node component potential is the weighted average of all node component potentials. The weights are assigned with respect to the relative importance of the attribute, e.g. in computational grid, compute potential is more important than disk potential. Further explanation is given in Appendix A.

3.1.3 Locally induced potential

The potential induced by a machine at point of attachment to the grid is the local grid potential induced by all machines attached to a node. Further explanation is given in Appendix A.

3.1.4 Grid potential

Grid potential is defined as maximum of local induced Grid potentials of all machines attached to the node.Further explanation is given in Appendix A. Thus, Two potential values are associated with each node (machine):

1. Potential that is induced by the local-machine's processing capabilities and

2. Potential that is induced at the local node due to remote nodes.

The Grid potential is determined by computing effective execution rate of the chosen benchmark code.

3.1.5 Remotely induced potential

The remotely induced potential have been proposed in [32]. As we move away from the node along the Grid, the potential induced at the point of attachment (node) decays. This decay rate in potential induced by a machine depends on the network characteristics. The remotely induced grid potential is the grid potential induced by a machine at a node other than its point of attachment.

The calculation is given in detail in Appendix A as discussed in [32]. The potential drop is determined by the execution time of the benchmark code on the source node and the communication cost (in terms of total latency) for transferring benchmark code source node to destination node. It is noted from the expression to calculate the potential drop given in Appendix A, that the communication cost and potential drop are inversely proportional. The pseudocode for determining the potential drop is described in Appendix A.

3.2 Data Dissemination Algorithms

The assumptions taken in the algorithms specified in the following subsections are as follows:

- A topology discovery is executed concurrently with the status dissemination algorithm. The topology of the network changes too slowly as compared to the status of resources. Thus, it is assumed here that the topology discovery algorithm is executed before the execution of the status dissemination algorithms.
- In the analysis of the algorithms, the rigorous proofs for the complexity expressions are not provided, instead an intuitive explanation is given.
- There is a lack of availability of the CPU load traces and query traces of any realistic computational grid deployed in the market.
- The change in the resource status depend on the interarrival time of the jobs and job size distribution. Here, the variation in the resource status is done according to the values drawn from the job size distribution . An exponential distribution is taken for job size distribution.

3.2.1 Universal algorithm

This algorithm is similar to the flooding algorithm used in the Internet Routers, i.e. Link state routing algorithm. In this algorithm, a node sends the status messages to every node it knows about, i.e. nodes in its neighbor list. In this algorithm, no filtering of the messages is done. Each node sends the resource status to every other node in the network.

3.2.1.1 Message Complexity

Let n be the number of nodes in the network, than to disseminate the status of a single resource, the message complexity of this algorithm becomes $n(n-1)$ for a single iteration.

3.2.1.2 Pros

It is the simplest kind of dissemination algorithm and each node contains accurate database of status of all nodes in the Grid. The variation of the flooding algorithm is still used to develop practical systems, e.g. Link state routing in Internet routers.

3.2.1.3 Cons

The Message complexity increases polynomially as number of nodes increases in the network. The accuracy comes at the cost of polynomial increase in the number of messages. This becomes significant when the scale of the grid increases.

3.2.2 Neighborhood algorithm

The algorithm in this group propagates the status information such that a node learns only about the nodes that are away from the fixed distance (in terms of hop-count) from it. This type of algorithm have been used in design of flock of condor pool system in [6], where condor pools are organized using the pastry [41] p2p overlay network. In this system, the messages are not only propagated to the nodes present in the pastry routing table, but also to the nearby condor pools using Time-to-live data. The distance (in terms of hop-count) is measured according to the shortest path from source to destination nodes. The detail algorithm described in [32] is given in Appendix A.

3.2.2.1 Message Complexity

Let n be the number of the nodes in the network and let k be the maximum degree. In neighborhood algorithm, each node sends messages to other nodes that lie within fixed network vicinity (in terms of hop-count). Let the radius of the vicinity is d. The number of nodes in the network vicinity is upper bounded by k^d. Since this expression is not dependent on n, the expression is constant. Thus, the message complexity of the neighborhood algorithm becomes $O(n)$ for the single iteration compared to $O(n^2)$ of universal algorithm. Please refer to Figure 3.2 for more clarification.

3.2.2.2 Pros

This is an attractive solution, since the probability that nearby nodes requires more accurate information rather than far off nodes is high. Here, the correlation between the application response time and the distance (in terms of hop-count) of the nodes from source node is assumed to be high.

3.2.2.3 Cons

- Use of Time-to-live does not always indicate the exact distance between two nodes. For example, consider a source node s and a destination node d. It is possible that either due to the congestion or collisions, a message gets dropped along the shortest path and another message reaches the node d through a longer route. In that case, the TTL would give a false estimate of the distance.

- The grid application schedulers may not be able to discover the powerful resources (in terms of computation) since the resource may not be located in the near vicinity of the requesting node. Due to this, powerful resources (in terms of computing) would not be utilized upto its maximum capacity.

3.2.3 Prioritized dissemination algorithm

In this algorithm, each node maintains two values 1. Self potential and 2. Average of remotely induced local potentials. As the node receives the status message (in terms of local potential) from remote nodes, it updates the average remotely induced potential and this message is further propagated to other nodes if the remotely induced local potential is greater than its local potential. The node sends the status message of its own resources only if the remotely induced potential is less than its average remotely induced local potential. This implies that a node that is in the vicinity of several powerful machines will not disseminate the messages since average remotely induced local potential will be large than its self potential. The pseudocode as described in [32] is given in Appendix A.

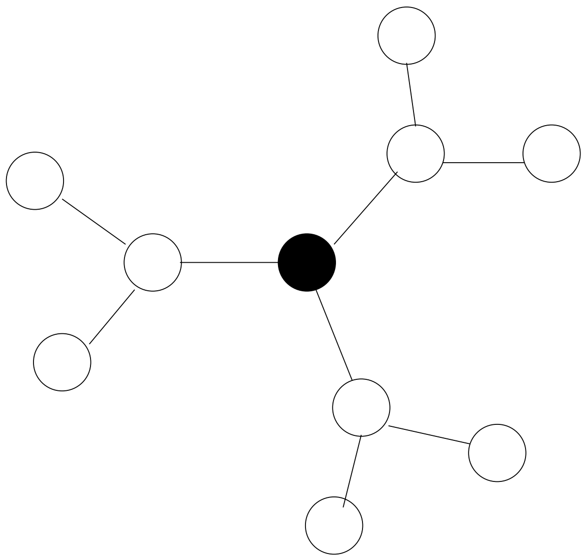

Figure 3.2: Number of nodes in the vicinity of the node (in dark) in neighborhood awareness algorithms, k=3, d=2

3.2.3.1 Message Complexity

The message complexity lies in between the universal algorithm and the neighborhood algorithm.

3.2.3.2 Pros

- By allowing significant nodes to transmit more frequently and more widely, will allow the grid application schedulers to discover the powerful resources present in the grid.
- The message overhead is significantly reduced compared to universal algorithm. The nodes that induce relatively less potential will not disseminate which will result into reduced dissemination error compared to neighborhood algorithm.
- As powerful resources disseminate more frequently and widely, these resources could be utilized up to its maximum capacity.
- The extent of the dissemination is dependent on the network link characteristics, which takes care of reducing the application response time.

3.2.3.3 Cons

The message complexity lies between neighborhood and universal algorithm.

3.2.4 Randomized dissemination algorithm

This algorithm is proposed in [32]. This algorithm uses the Grid potential concept as the prioritized algorithm discussed in the previous subsection. Let the self potential of a node be Φ_s, and the remote induced potential at the node be Φ_r. The node sends status messages with probability $p = 1 - e^{-(\Phi_s - \Phi_r)/m}$ to other nodes, where m is a constant. Because each neighboring node is selected with a probability p and each node has $n - 1$ neighboring nodes, the expected number of nodes to receive the status message is $E(p \times (n-1))$. This algorithm is very similar to Name-Dropper algorithm in [22] except here the Grid potential is used to control the number of status messages emitted into the network such that more significant nodes transmit more messages than less significant nodes. The pseudo-code for this algorithm is presented in Appendix A as described in [32].

3.2.4.1 Message Complexity

The randomized algorithm can take more than a round to converge as compared to only single round needed in neighborhood and universal algorithm. The randomized is similar to Name-Dropper algorithm proposed in [22], which is shown to converge in $O(log^2 n)$ rounds with high probability.

3.2.4.2 Pros

All advantages of Prioritized dissemination algorithm are achieved in this algorithm. Also, the algorithm may restrict the message propagation even though the remotely induced potential is greater than the local potential, thus further reducing the message complexity.

3.2.4.3 Cons

The selection of the probability is a caveat and must be tested so as to yield the desired results. In the next chapter, the details of simulation setup in ns-2 are discussed. The performance parameters being taken into consideration for above algorithms and simulation results are discussed in the chapter 5.

Chapter 4

Peer-to-Peer Network Simulation in ns-2

4.1 Peer-to-Peer networks simulation

We need a simulation software to evaluate the P2P protocols. Making live tests of P2P protocols is problematic. This is because, the hardware and software components required to enable P2P systems are lacking. There are several P2P simulation softwares available in the market, but they are in some or other way developed for evaluating specific P2P systems like DHTs, which is not desirable. These simulation softwares are not suitable to simulate the new P2P protocols, since the functionalities like

- Support to design new test cases for simulation
- Analysis of the trace files
- Developing new P2P protocols

may not be included. In the following subsection, the required characteristics of a simulation software for the problem are enlisted and several simulation softwares are investigated with respect to these characteristics.

Requirements of a simulator for peer-to-peer systems simulation:

1. It should be a discrete event based open-source simulator.
2. It must support the simulations of Transport layer protocols & Unicast routing algorithms.
3. It should have the implementations of various Queue management mechanisms.
4. It should support topology generation.
5. It must support the trace file generation.

6. It should include the support to extend the functionality of the Transport layer agents, applications, routing algorithms and protocols of transport layer, node and packet.

7. It should have a visualization tool to support the analysis.

8. It should have a trace analysis tool to figure the complex parameters, like number of packets send, forwarded, dropped, links in which packets have been dropped.

9. It must support large-scale simulations.

10. Object-oriented design is desired, so that an extension to the simulator becomes easy.

4.2 Analysis of simulation softwares

4.2.1 Simjava

Simjava [34] [24] is a process based discrete event simulation package for Java [3], similar to Jade's Sim++ [14], with animation facilities. Simjava provides only the basic set of APIs for discrete event scheduling, and does not provide the implementation of transport layer and network layer protocols. Thus, this simulator is not suitable for peer-to-peer network simulation.

4.2.2 Planetsim

PlanetSim [20] is an object-oriented simulation framework for overlay networks and services. Planetsim includes the support for structured peer-to-peer networks. Thus, it becomes difficult to implement an unstructured peer-to-peer network.

4.2.3 SSFNet 2.0

SSFNet [10], is suitable for peer-to-peer simulation, but to run the simulation, it is required to learn the new modeling language known as DML.

4.2.4 PeerSim

Peersim [36], does not allow to specify the customized routing algorithm or transport layer protocol.

4.2.5 p2psim

p2psim [42] already supports Chord [35], Accordion [27], Koorde [28] , Kelips [21], Tapestry [46], and Kademlia [33]. These implementations are specific to p2psim. They consist of substantially fewer lines of code than the real implementations.

p2psim includes the support for structured peer-to-peer networks. Thus, it becomes difficult to implement an unstructured peer-to-peer network.

4.2.6 Comparison of the characteristics of simulators

The comparison table 4.1 and 4.2 of characteristics of several discrete-event simulators is presented below:

Simulation tool	Discrete Event Scheduler	Open Source	Widely used and large user community	Object-oriented Design	Support for new protocol
SimJava	Yes	Yes	No	Yes	Yes
PlanetSim	Yes	Yes	No	Yes	Very low
OPNET	Yes	No	Yes	Yes	Very low
SSFNet	Yes	Yes	Very low	Yes	Yes
PeerSim	Yes	Yes	No	Yes	No
p2psim	Yes	Yes	No	Yes	Very low
ns-2	Yes	Yes	Yes	Yes	Yes

Table 4.1: Comparison of discrete-event simulators

Simulation tool	Advance network functionalities	Topology Generation	trace analysis	large-scale simulations	Visualization tool
SimJava	No	No	No	Very low	No
PlanetSim	Very low	Yes	No	Very low	No
OPNET	Yes	Yes	Yes	Yes	Yes
SSFNet	Yes	Yes	Very low	Yes	Yes
PeerSim	No	No	No	Yes	No
p2psim	No	No	No	Yes	No
ns-2	Yes	Yes	Yes	Yes	Yes

Table 4.2: Comparison of discrete-event simulators

4.3 P2P simulation in ns-2

In this section stages of development of P2P simulator for ns-2 are presented.

The Grid can be considered as a virtual network of various systems that are interconnected by high-speed links. Also, it is observed that just like the Internet routing, the nodes in the Grid is grouped into "autonomous systems" called "Grid domains". The nodes in the same Grid domain have same resource management and security policies. The nodes in the Grid can be of two types:

Endsystems: The nodes from which the series of resource requests are initiated and nodes which provides the resources.

Router-allocators: The nodes which are responsible to route the resource requests generated by Endsystems. The Router-allocators may route the request to another Router-allocator or an Endsystem. The Router-allocator sits on the application layer of the TCP/IP stack.

In this thesis, for the purpose of modeling the resource discovery schemes, the Grid is modeled as a connected graph. A node of the graph corresponds to both a Router-allocator and an Endsystem. An edge in the graph corresponds to a path between two Router-allocators or path between a Router-allocator and an Endsystem. To simplify the analysis, each node is assumed to behave as both an Endsystem and a Router-allocator.

A data dissemination algorithm running on any node of a Grid should perform the following functions:

- Receive the messages that are destined for the node
- Relay messages that are destined to other nodes
- Generate messages to other nodes on behalf of the node

Thus, need to make some changes to the ns-2 simulator to fulfill the above requirements arises.

4.3.1 P2P Simulator Architecture

To develop Peer-to-Peer extension for NS-2 new P2P agent was created and some modules (agent and routing) were extended. The application consists of two major components:

- P2P agent
- Extension of the routing module

Each P2P agent is attached to corresponding node and acts on transport level of OSI model. Each agent has receiving and sending procedures. Sending procedure initializes each packet and sends it to the environment. Receiving procedure is used for processing received packets.

The P2P agent can be easily extended for adding new capabilities, such as supporting of new protocols and redesigned version of old protocols.

4.3.2 Packet

The ns-2 packet is composed of a stack of headers, and an optional data space (see Figure 4.1). A packet header format is initialized when a Simulator object is created, where a stack of all registered (or possibly useable) headers, such as the common header that is commonly used by any objects as needed, IP header, TCP header, RTP header (UDP uses RTP header) and trace header, is defined, and the offset of each header in the stack is recorded. What this means is that whether or not a specific header is used, a stack composed of all registered headers is created when a packet is allocated by an agent, and a network object can access any header in the stack of a packet it processes using the corresponding offset value.

The packet header is modified as per the requirements of the simulation. The figure 4.1 shows the details of the packet header fields that are added. The field named "l_grid_potential_" is used to store the locally induced compute potential. Other fields are self-explanatory.

4.3.3 Agent

Agents represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers. Agents can be considered as the processing elements of packet attached to a node. The "Agent" class of ns-2 is extended to include the functionalities enlisted in section 4.3. Figure 4.2 shows the hierarchy of inheritance of classes in ns-2 and relationship of these with P2PAgent class and other algorithm specific peer-to-peer agents. Figure 4.3 shows the details of "P2PAgent" class and relationship with other classes. The P2P Agent is attached to every node in the topology. Depending on the dissemination algorithms, the P2Pagent is extended to include the functionality. Thus, following agents are created:

- Universal Agent: For Universal Awareness algorithm
- Neighborhood Agent: For Neighborhood Awareness algorithm
- PDP Agent: Prioritized dissemination algorithm

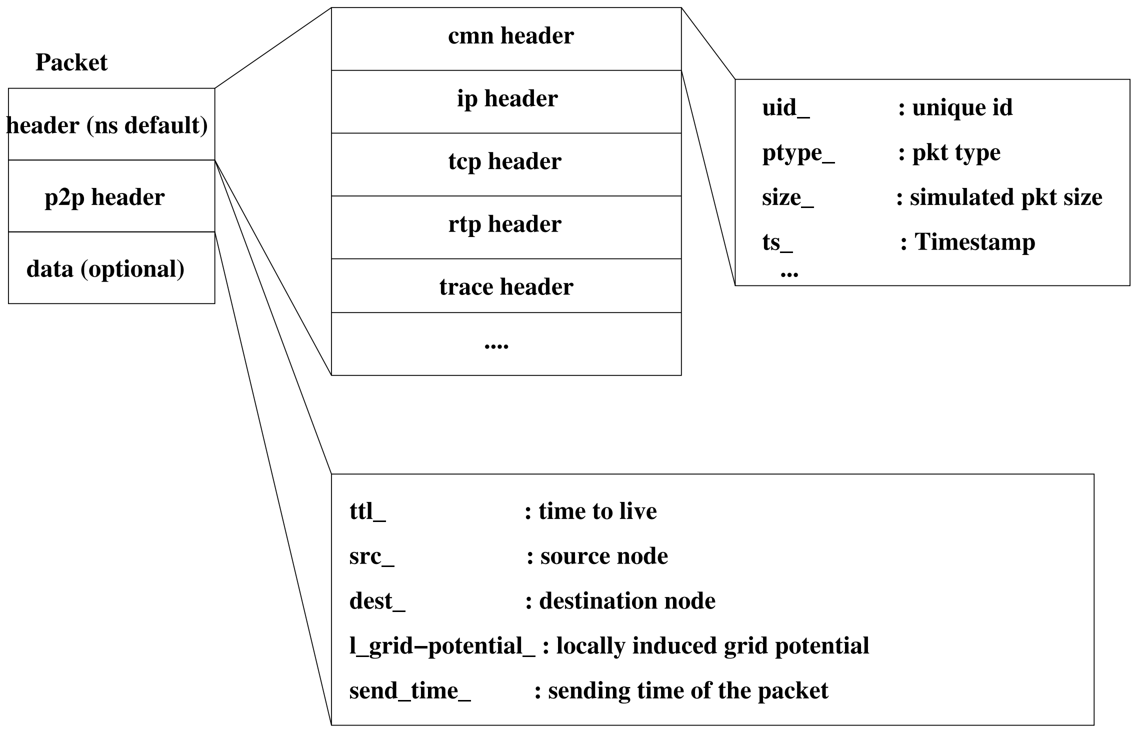

Figure 4.1: Packet header format in ns-2

- ND Agent: Randomized dissemination algorithm

P2P Agent also generates the traffic similar to CBR, since packets of fixed size are generated at regular intervals. The traffic generated by this, becomes similar to CBR if observed for long interval of time. Thus, each node consists of P2PAgent and in turn, p2pagent contains the CBR traffic generator. This is depicted in Figure 4.4.

4.3.4 Routing Module

In ns-2, a routing agent of a node would intercept the packet generated by the transport agent on that node, and the packet is routed to a particular destination given by the transport agent of that node. It is required that every packet that is forwarded, is processed. Also, no changes has been made in routing agent; we need to pass on every routed packet to the corresponding P2P Agent (transport layer agent) of the corresponding node. The Figure 4.6 depicts the desired functionality. Also, the changes made to the RouteLogic class of ns-2 to implement this functionality are given in detail in Figure 4.5.

4.3.5 Topology generation

The GT-ITM (Georgia Tech - Internet Topology Modeling) described in [45], [8] and [44],

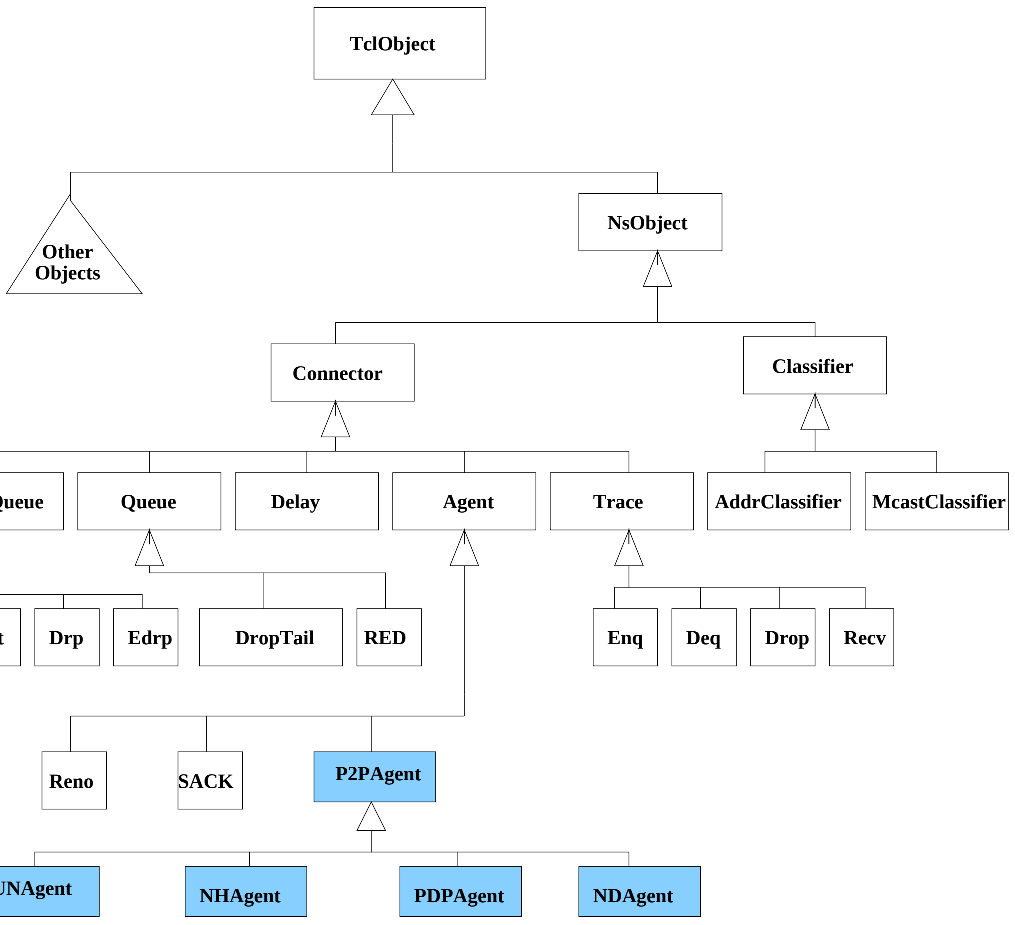

Figure 4.2: Class diagram of ns-2

is a topology generation tool used to generate the topology for the overlay network. The GT-ITM topology generator can be used to create flat random graphs and two types of hierarchical graphs, the N-level and transit-stub. The output format of the graphs generated by gt-itm is in sgb (Stanford graphics base) format specified by Don knuth. Another program known as “sgb2ns” is required to convert this into tcl format so that it can be used in ns-2 simulation. Also, to get the scalar as well as vector quantities [45] specifying the characteristics of graphs generated by gt-itm, the program named as “edriver” is provided.

4.3.6 Trace Analysis

The trace analysis is done with the tool named as “tracegraph” [4]. This tool is recently been developed for the exhaustive analysis of ns-2 trace files. The information like simulation information (number of packets forwarded, number of lost packets, number of dropped packets, number of sent packets), End2End delays, simulation round trip times, simulation processing times at intermediate nodes, delays between current node and other node, wired as well as wireless trace analysis support, 2D graphs (Throughput of receiving packets at current node, Throughput of forwarding packets at current node, Throughput of dropping packets at current node etc.), 3D graphs (Number of forwarded packets at all the nodes, Numbers of received packets at all the nodes, Number of dropped packets at all the nodes etc.), graphs like packet size vs. jitter, packet size vs. processing time, packet size vs. RTT, packet size vs. throughput etc., are provided. This tool has also got a user-friendly graphical user interface.

4.4 Summary

In this chapter, the selection of appropriate simulation tool and comparison of several simulation tools was presented, according to several criteria defined in this chapter. The simulation setup and details of extensions done to ns-2 simulation tool was also presented. The class diagrams for implementation of various transport agents were also presented. The next chapter discusses the simulation parameters for simulation and interpretation of the simulation results.

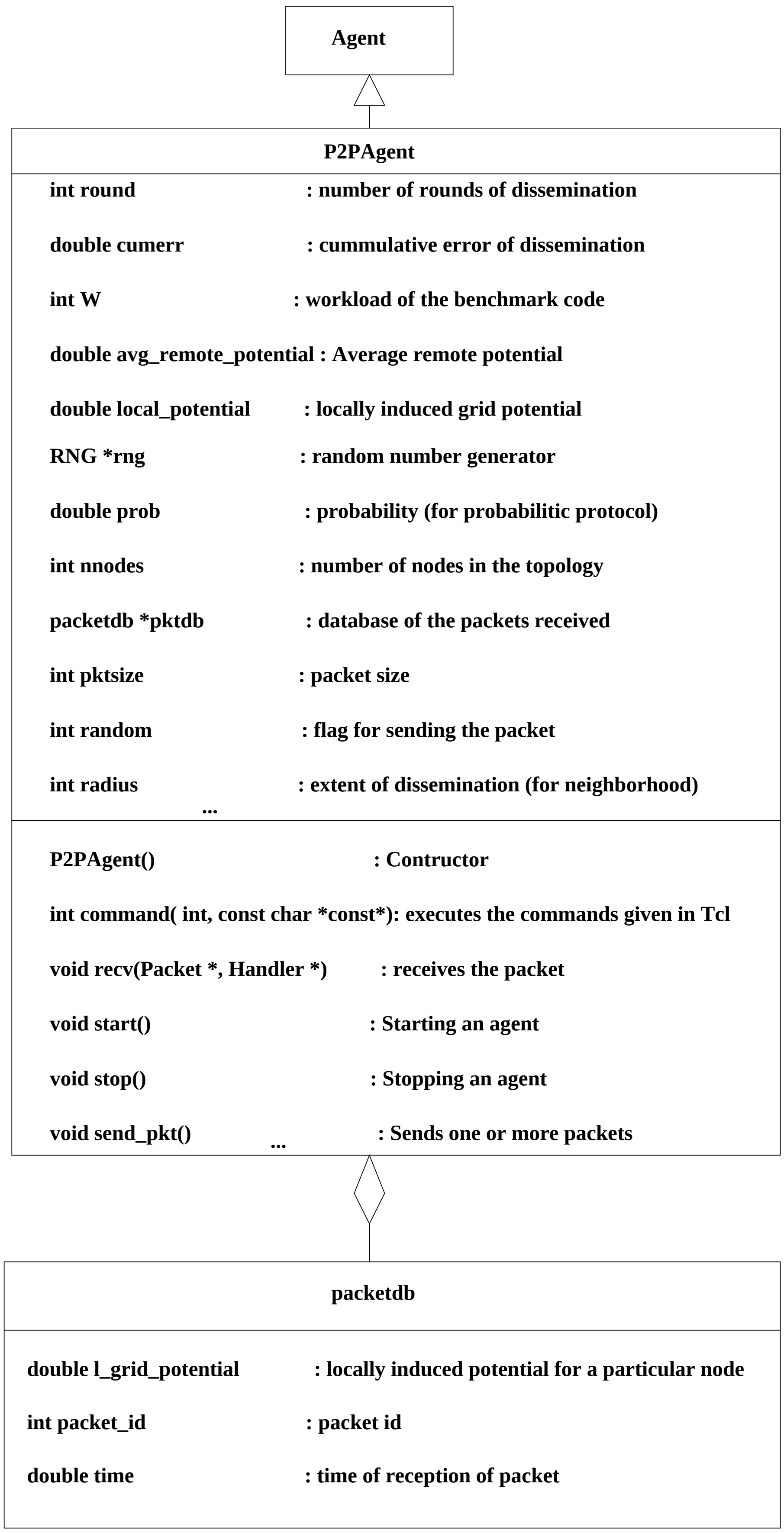
Agent
P2PAgent
int round : number of rounds of dissemination
double cumerr : cummulative error of dissemination
int W : workload of the benchmark code
double avg_remote_potential : Average remote potential
double local_potential : locally induced grid potential
RNG *rng : random number generator
double prob : probability (for probabilitic protocol)
int nnodes : number of nodes in the topology
packetdb *pktdb : database of the packets received
int pktsize : packet size
int random : flag for sending the packet
int radius : extent of dissemination (for neighborhood)
...
P2PAgent() : Contructor
int command(int, const char *const*): executes the commands given in Tcl
void recv(Packet *, Handler *) : receives the packet
void start() : Starting an agent
void stop() : Stopping an agent
void send_pkt() : Sends one or more packets
...
packetdb
double l_grid_potential : locally induced potential for a particular node
int packet_id : packet id
double time : time of reception of packet

Figure 4.3: Class diagram containing P2PAgent class and relationship with other classes

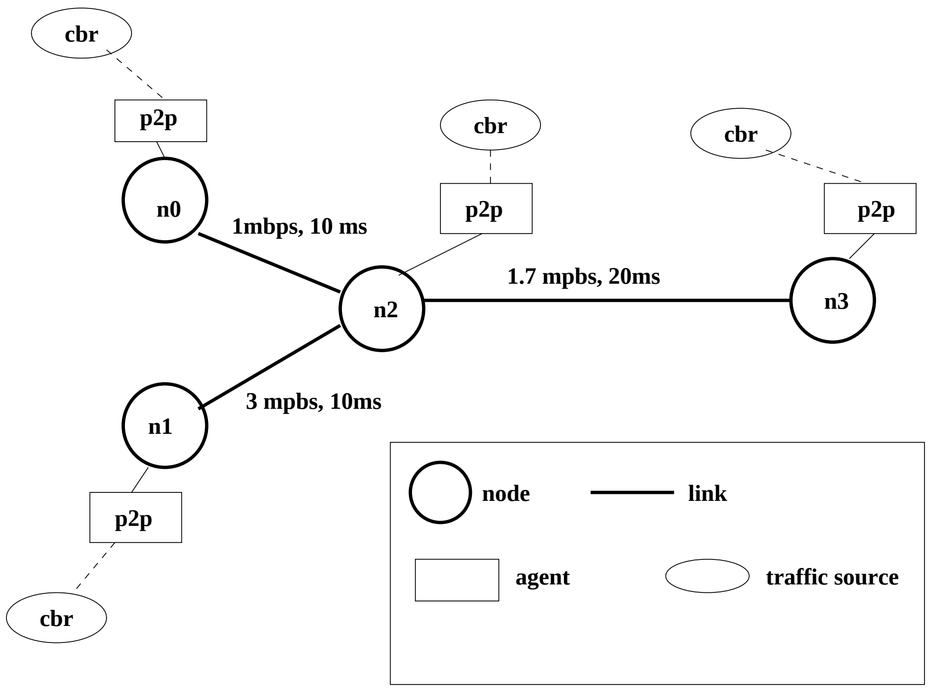

Figure 4.4: Traffic source, agents and nodes in the p2p network setup

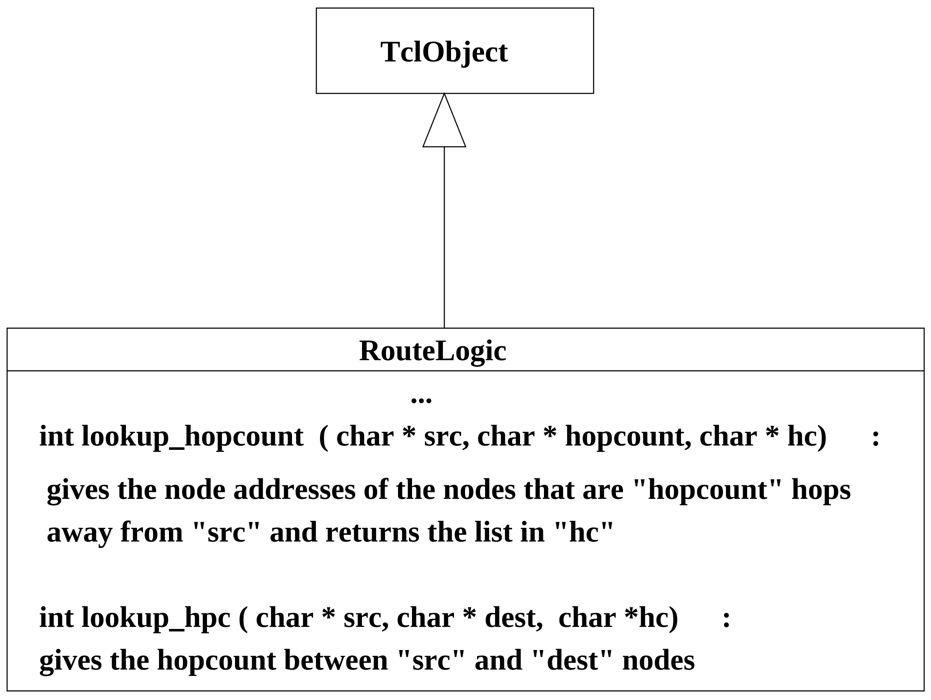
TclObject
RouteLogic
...
int lookup_hopcount (char * src, char * hopcount, char * hc) :
gives the node addresses of the nodes that are "hopcount" hops
away from "src" and returns the list in "hc"
int lookup_hpc (char * src, char * dest, char *hc) :
gives the hopcount between "src" and "dest" nodes

Figure 4.5: Class diagram containing RouteLogic class and relationship with TclObject class

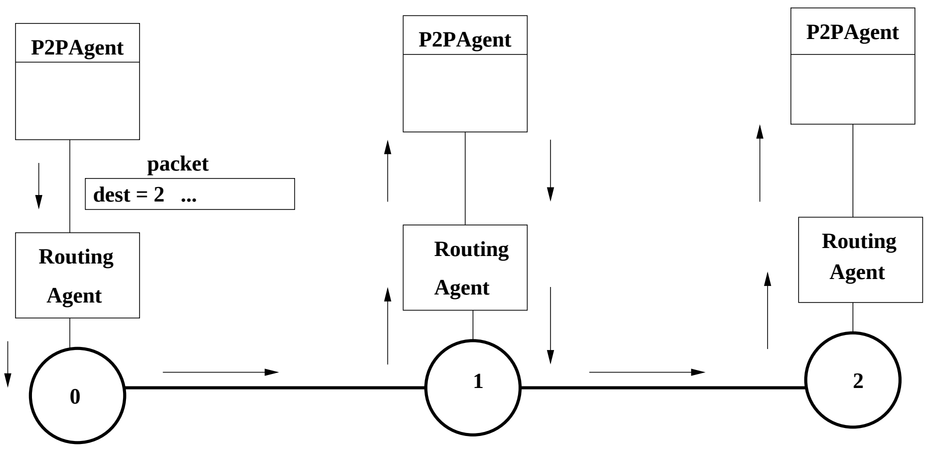

Figure 4.6: Desired functionality for P2P Agent

Chapter 5

Experiment setup and Results

This chapter includes the simulation experiment setup in which, the parameters taken into consideration for performance evaluation of the information dissemination algorithms are discussed. The results and the analysis of the results are also discussed in this chapter.

5.1 Simulation experiment setup

To evaluate the performance of various information dissemination schemes, the following simulation study is devised.

In this simulation study, a computational Grid is modeled by a random graph with the nodes denoting the machines. The information dissemination scheme is responsible for updating the status database that is maintained at each node. Depending on the scheme that is under consideration, there might be a complete database at each node or an incomplete database at each node.

5.1.1 Dissemination Error

The parameter to measure the accuracy of the information each node consists of, is defined by dissemination error. The Dissemination error defined in [32] is as follows:

Dissemination error is defined as the difference between a repository's/node's local view of another repository's data and the actual value of that repository's data. A view is the latest data that one repository has about another. Let $V(R_{ij})$ denote repository $R_i's$ view of repository $R_j's$ data, and let n be the total number of repositories within an overlay network. The absolute error for R_i repository is given below:

$E_i = 1/n \times \sum_{j=1}^{n} |V(R_{ij}) - V(R_{jj})|$

$n = number of repositories/nodes$

5.1.2 Message Complexity

The dissemination efficiency as well as dissemination error will depend on the number of messages exchanged. Thus, another performance parameter used is message complexity. The message complexity depends on the degree distribution of a given physical topology. The message complexity is calculated as below:
$Messagecomplexity = Num.ofmessagesgenerated + Num.ofmessagesforwarded$

The number of messages forwarded depends on the degree distribution of given physical topology. As the physical topology graph tends to become a complete graph, the number of forwarded messages will decrease. Thus, the physical network topology will affect the performance of the dissemination algorithms.

5.1.3 Grid potential distribution

This refers to the local potential values taken for the simulation. These values are taken from the GridG artificial grid annotator [31]. The reasons behind selecting GridG as the synthetic workload generator for simulation are as follows:
Smith et al. [12] studied the update and query processes on grid information, there is no extant work and limited available data on the structure of computational grids makes difficult to model a computational grid and generate the test data for workload for simulation purpose. Although the correctness of the protocols does not depend on the underlying topology, their efficiency does depend on the topology. Unfortunately, very little is known about the characteristics of a grid or network that are represented by the annotations. GridG produces the topology and annotates the topology with host attributes with user-supplied distributions and correlations between various host attributes. Some of these correlations are between Architecture of the host and Operating system, Architecture and CPU MHz, Architecture and Number of CPUs, Number of CPUs and Memory, Number of CPUs and disk etc. Also, the correlations between attributes on nearby hosts are also taken into account to annotate the grid topology.

Thus, GridG is used to generate the initial local potential values to be assigned to each node. The heterogeneity in the grid affects the performance of the protocols e.g. if the grid consists of heterogeneous nodes (in terms of processing power), the performance of the protocols will increase. And this assumption is valid since; the grid consists of mainly the heterogeneous computational nodes. Due to this, we test the protocol performance with relatively homogeneous compute resources and heterogeneous resources. We derive the potential values from uniform distribution for homogeneous compute resources. The data is the product of Processing power of CPU (in terms of MHz) and Number of CPUs in a node. The values derived for Processing power (in MHz) is in range of 8-2500 MHz and the values derived for Number of CPUs in a ndoe is in range of 1-128. These values are derived using the rules in GridG annotator. The rules states the percentage of the each type of CPUs (in terms of MHz) and type of nodes (in terms of number of CPUs).

5.1.4 Potential drop

As discussed in section 3.1.5 in chapter 3, the locally induced compute potential gets decayed as the node gets farther from the launch point of the application. Thus, we need to decide on the rate of decay of local potential of individual node. The selection of the amount of drop of the local potentials is a caveat. But, it is clear that if the decay is low then the message complexity will be high and message complexity will be low if decay is high.

The performance of four protocols is measured with the parameters discussed above.

5.2 Experiments and Results

All protocols are executed for 4 iterations. The values plotted in the graphs are average of these values. The average dissemination error is calculated by taking the mean of the dissemination errors obtained by taking the difference between the dissemination error of the protocols and dissemination error of swamping algorithm. The results plotted for randomized protocols are the average of 3 runs of the protocols with same parameter setting. In addition to this, the randomized protocols are analyzed by varying the value of k ($k = 1/m$, where m is a constant term in the probabilility calculation expression in section 3.2.4). As the value of m increases, the difference between remotely induced potential and local potential will be scaled down higher and thus allowing the high probability to be assigned for the destination nodes who induce the potential of relatively higher magnitude than the local potential. The selection of value of m depends on the mean and standard deviation of the grid potential distribution in a grid.

5.2.1 Experiment 1: Performance analysis of protocols in 10-node heterogeneous grid

In this experiment the performance of all four protocols is measured when executed on 10-node heterogeneous grid. The topology generation method used for generating the 10-nodes topology is pure random. All protocols are tested on different topology configurations. The configurations are shown in table 5.1. In this experiment, it has been shown

Sr. No.	Alpha	Scale	Bicomponents	Diameter(hop-count)	Average Degree
1.	0.3	10	5	6	1.9
2.	0.5	10	2	3	3.2
3.	0.8	10	1	2	6.8

Table 5.1: Three different topology configurations for 10-node topology

that performance of the protocols depends on the local potentials distributed to the Grid nodes. Two types of data sets for generating the artificial local potentials are used to evaluate the performance, homogeneous and heterogeneous data sets. In homogeneous data set, the values are drawn from uniform distribution. In heterogeneous data sets, the data set with high variance is taken. The data set shown in Figure 5.1 has mean=251.40 and standard deviation=754.95, which shows the high variance in the data. In this experiment, the heterogeneous data set is taken into consideration in which diverse set of computational resources are present in grid. The diversity is with respect to processing power(in MHz). The values of local potentials are generated through GridG synthetic grid annotator. The input parameter given to GridG annotator is the percentage of high performance computing machines like clusters containing large number of CPUs with high performance. The spike in the local potential data set plotted in Figure 5.1 indicates that a cluster (with large number of CPUs) is attached to the Grid. The GridG uses certain heuristics for Intra host and Inter host configurations e.g.

1. One processor will have memory between 64M and 4G.
2. More CPUs, more likely to have bigger memory disk
3. More memory, more likely to have bigger disk and vice versa
4. Machines with different architectures have different distributions of CPU clock rate.

The values plotted in Figure 5.1 represents the number of CPUs $\times$ operations/sec.
Results: The performance of the protocols is measured with respect to Message Complexity and Dissemination error. The Figure 5.2 shows the Average Message overhead across all protocols and Figure 5.3 shows the Average Dissemination error across all protocols.
Observation: From the results, it is noted that significant performance is achieved by PDP and Randomized protocols with comparable message complexity with Neighborhood protocol. It is observed in the Figure 5.2 that message overhead of swamping is the highest and corresponding error in Figure 5.3 is the lowest. Inspite of the message overhead of

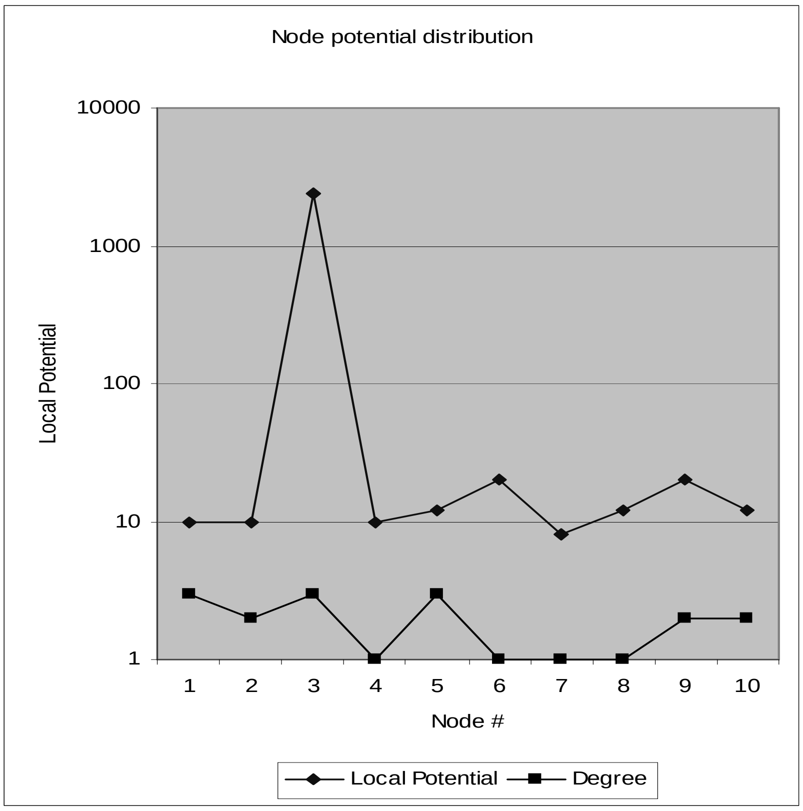

Figure 5.1: Distribution of Local potentials to the nodes in the grid and corresponding degree of the nodes in 10-node heterogeneous grid.

PDP protocol being lesser than Neighborhood protocol, the error of PDP protocol is significantly less than Neighborhood protocol. The Randomized protocols with different values of k are analyzed. It is observed in the Figure 5.2 that the message overhead of Randomized protocol with $k = 0.5$ is higher than other randomized protocols since, the small changes (magnitude of 2.0) in the potentials of nodes known to the particular node are filtered or are assigned very low probability, thus increasing the message overhead and reducing the corresponding error significantly as shown in Figure 5.3. The randomized protocol with $k = 0.02$ achieves highest performance compared to all other protocols since the message overhead is relatively less compared to all other protocols and dissemination error is also less compared with randomized protocol with $k = 0.07$, PDP and Neighborhood protocol.

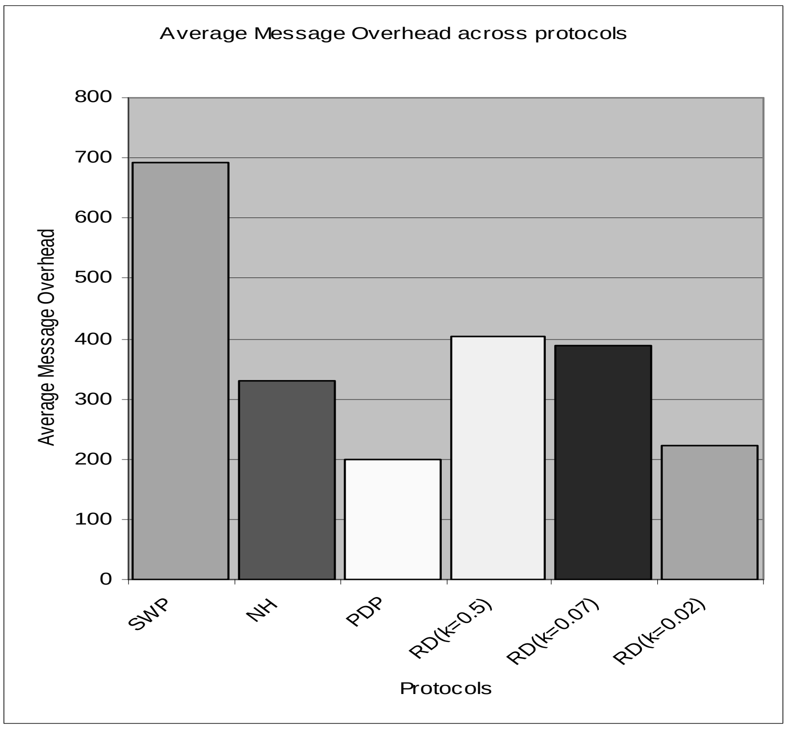

Figure 5.2: Average Message Overhead across all protocols for 10-node heterogeneous grid.

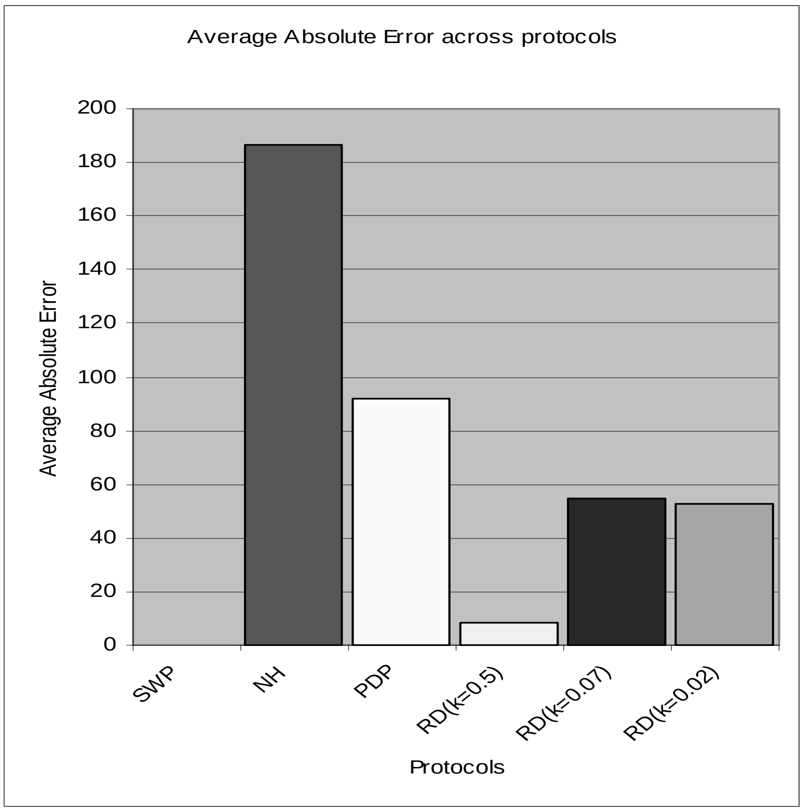

Figure 5.3: Average Absolute Dissemination Error across all protocols for 10-node heterogeneous grid.

5.2.2 Experiment 2:Performance analysis of protocols in 10-node homogeneous grid

In this experiment the performance of all four protocols is measured when executed on 10-node homogeneous grid. The topology generation method used for generating the 10-nodes topology is pure random. All protocols are tested on different topology configurations. The configurations are shown in table 5.1. The Grid potential distribution is plotted in Figure

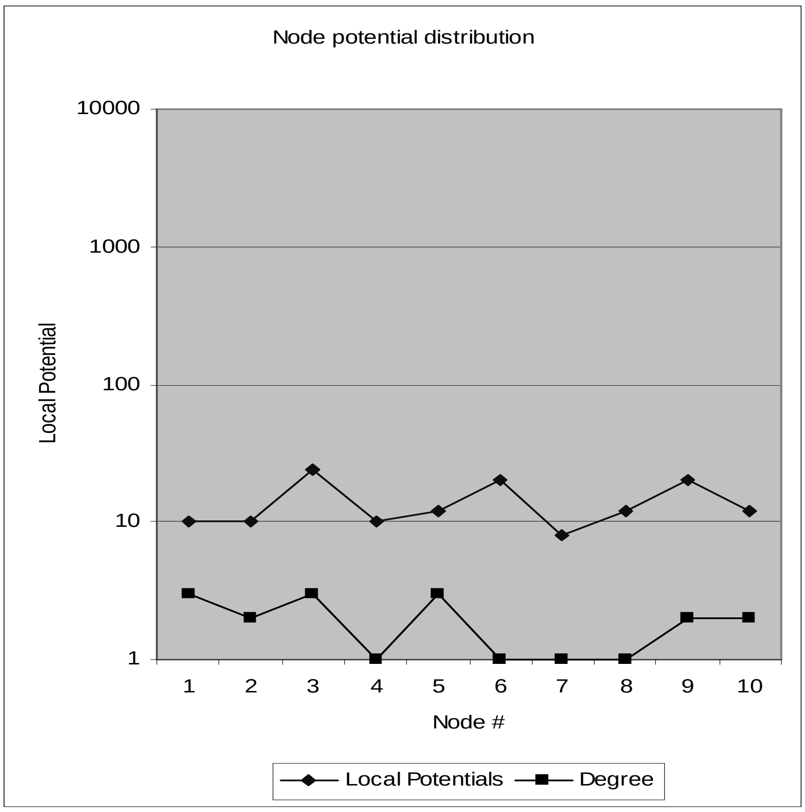

Figure 5.4: Distribution of Local potentials to the nodes in the grid and corresponding degree of the node (homogeneous grid) for 10 nodes

5.4. The Figure 5.4 shows that the values of local potentials are approximately uniform.

Results:The Figure 5.5 shows the Average Message overhead across all protocols and Figure 5.6 shows the Average Dissemination error across all protocols in a homogeneous grid.

Observation:
In this experiment, the homogeneous grid is considered in which the local potential values are drawn from uniform distribution. It is observed from Figure 5.5 and Figure 5.6 that

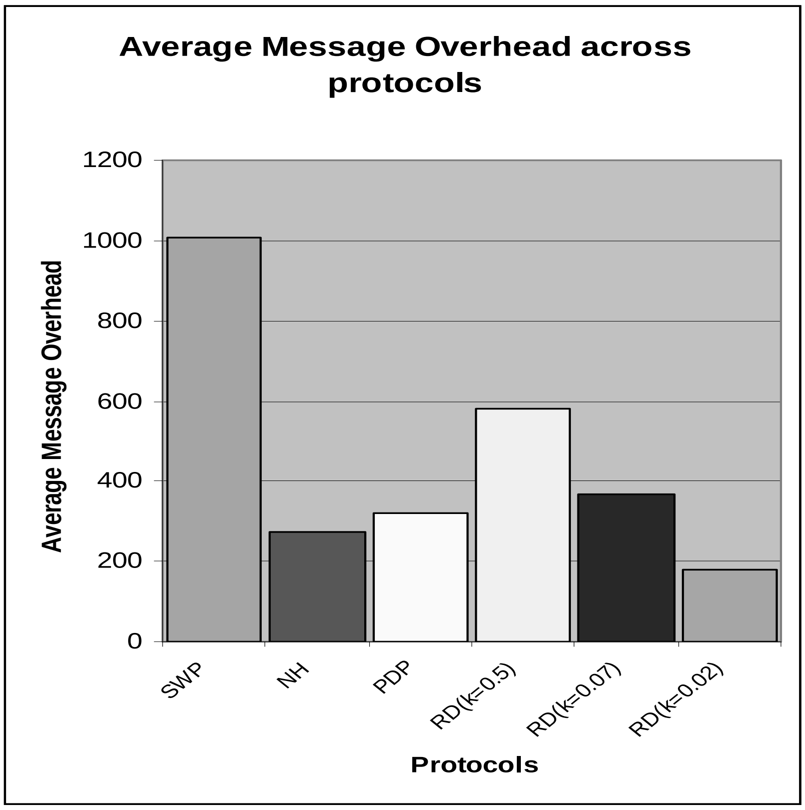

Figure 5.5: Average message overhead across all protocols for 10-node homogeneous grid

Performance of PDP is approximately same with Neighborhood protocol with message complexity of Neighborhood and PDP approximately same as shown in Figure 5.5. Thus, PDP behaves as Neighborhood protocol in case of homogeneous grid. It is further observed that message overhead of randomized protocols ($k = 0.07$ and $k = 0.02$),Neighborhood and PDP protocols are approximately same and yielding approximately same dissemination error.The mean of the distribution is 13.8 and standard deviation of 5.4528 which is a less varying data set compared to the data set considered in heterogeneous grid. Thus, it can be concluded that the PDP and randomized protocols performs better than Neighborhood protocols in case of heterogeneous grid which is a typical scenario in grid.

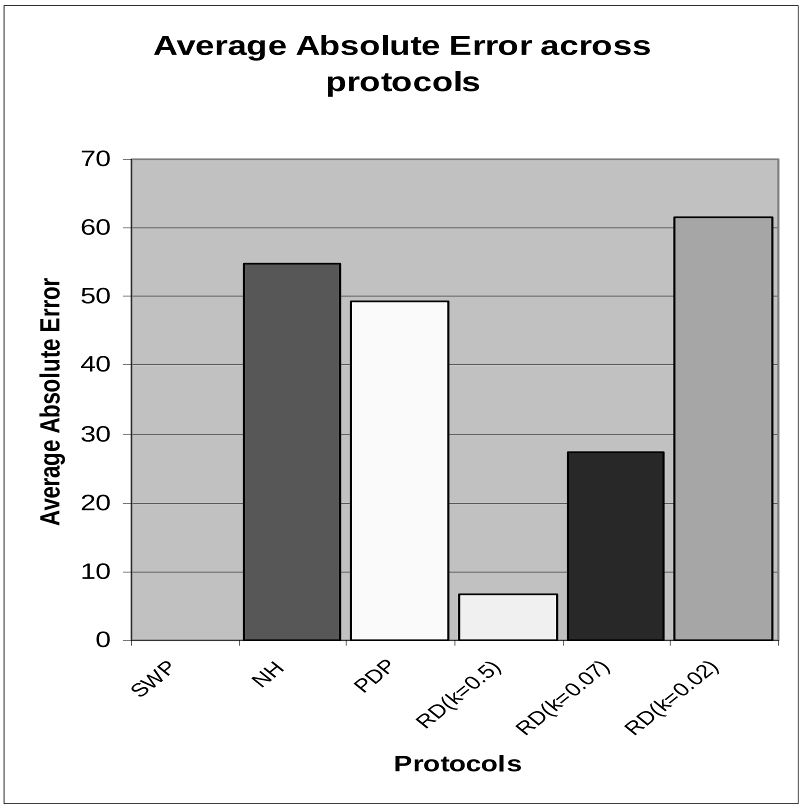

Figure 5.6: Average dissemination error across all protocols for 10-node homogeneous grid

5.2.3 Experiment 3: Performance analysis of protocols in 50-nodes heterogeneous grid

In this experiment the performance of all four protocols is measured when executed on 50-node heterogeneous grid. The topology generation method used for generating the 50-nodes topology is pure random. All protocols are tested on different topology configurations. The configurations are shown in table 5.2. The data set shown in Figure 5.7 has

Sr. No.	Alpha	Scale	Bicomponents	Diameter(hop-count)	Average Degree
1.	0.09	50	4	7	3.92
2.	0.3	50	1	3	14.20

Table 5.2: Two different topology configurations for 50-nodes topology

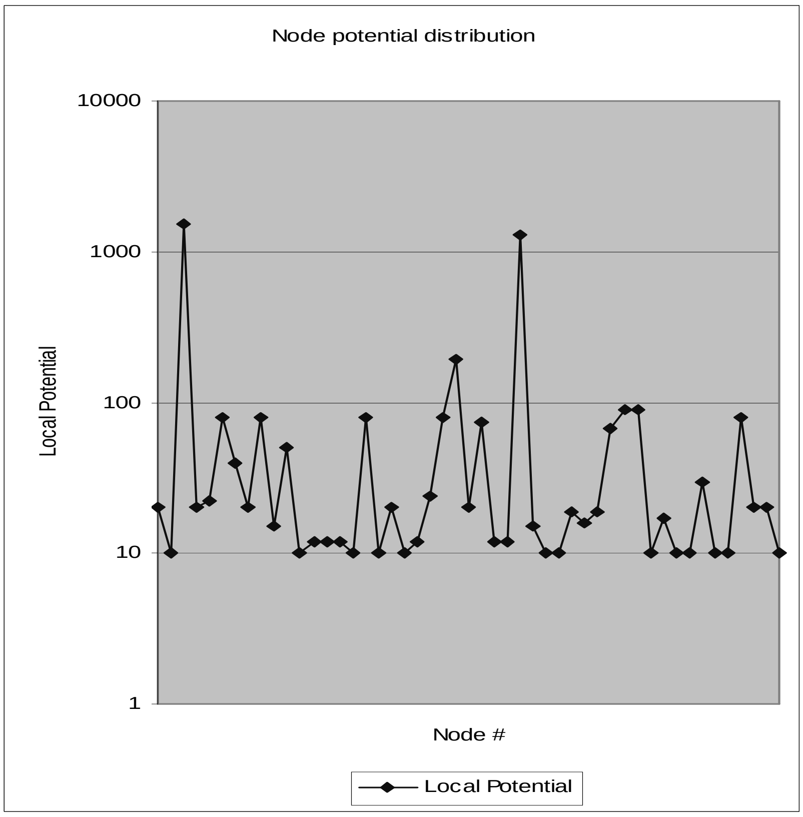

Figure 5.7: Distribution of Local potentials to the nodes in the heterogeneous grid for 50 nodes

mean of 88.40 and standard deviation 278.414, which shows the high variance in the data.

In this experiment, the heterogeneous data set is taken into consideration in which diverse set of computational resources are present in grid.

Results:The Figure 5.8 shows the Average Message overhead across all protocols and Figure 5.9 shows the Average Dissemination error across all protocols in a 50-nodes heterogeneous grid.

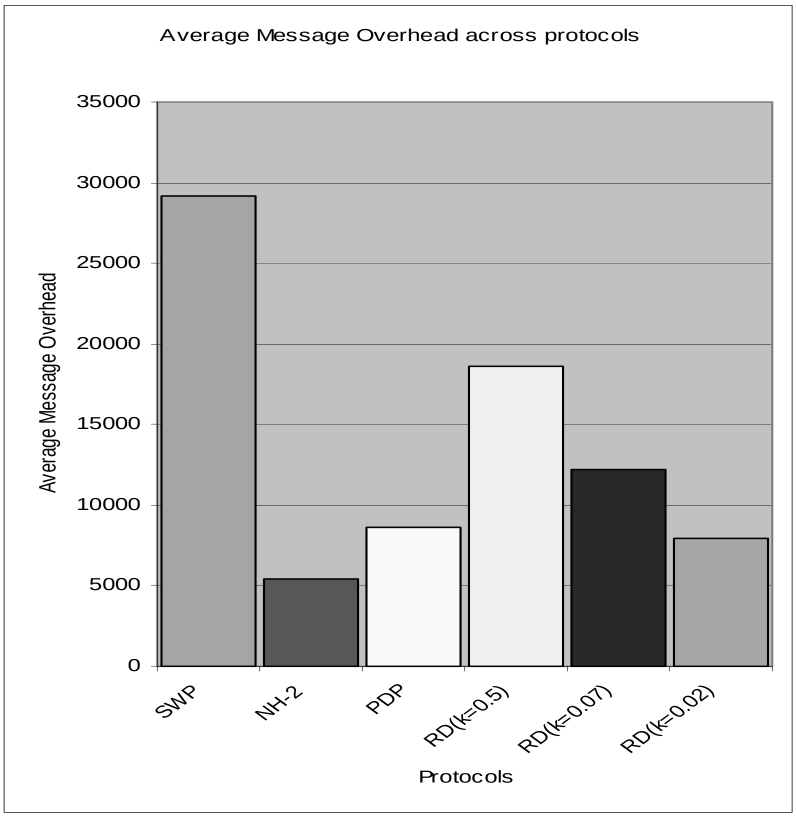

Figure 5.8: Average message overhead across all protocols for 50-nodes heterogeneous grid

Observation: It is observed from the Figure 5.8 and Figure 5.9 that the message overhead of Neighborhood protocol and randomized protocol(k=0.02) are approximately same, but the error of neighborhood is significantly high as compared to randomized protocol(k=0.02) and randomized (k=0.07). Also, slight increase in message overhead as compared to neighborhood, randomized protocol (k=0.5) also performs better than neighborhood protocol. The randomized protocol (k=0.07) and randomized protocol (k=0.02) out performs PDP protocol also.

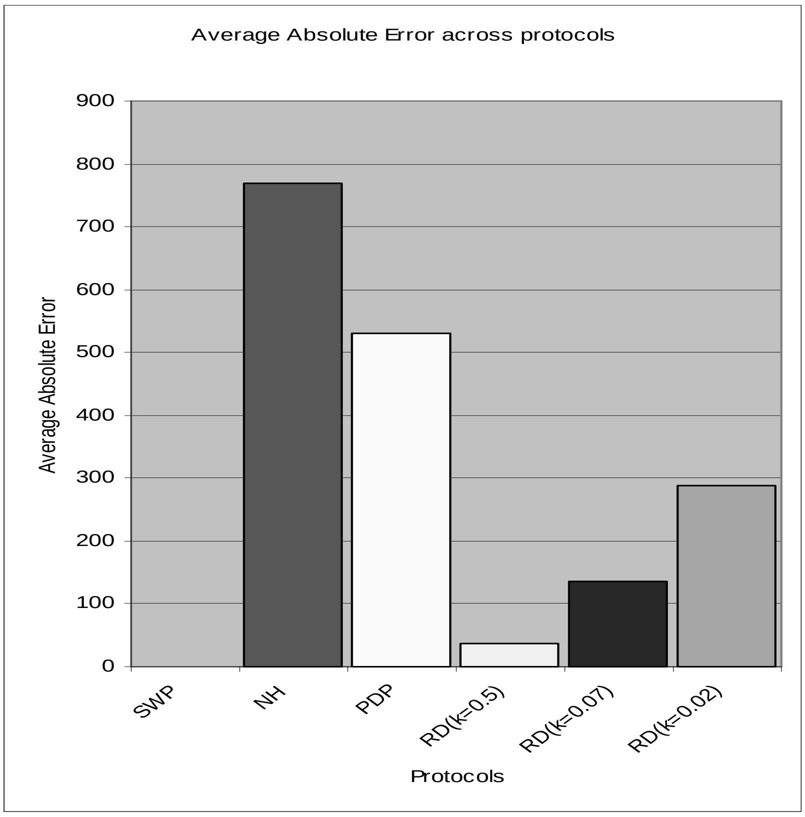

Figure 5.9: Average dissemination error across all protocols for 50-nodes heterogeneous grid.

5.2.4 Experiment 4: Performance analysis of protocols in 100-nodes heterogeneous grid

In this experiment the performance of all four protocols is measured when executed on 100-nodes heterogeneous grid. The topology generation method used for generating the 100-nodes topology is pure random. The configuration is shown in the table 5.3. The

Sr. No.	Alpha	Scale	Bicomponents	Diameter(hop-count)	Average Degree
1.	0.03	100	13	9	3.16

Table 5.3: A topology configuration for 100-nodes topology

data set shown in Figure 5.10 has mean of 369.13 and standard deviation 2377.918, which shows the high variance in the data. The variance of this data set is larger than the data set taken in the earlier simulations. In this experiment, the heterogeneous data set is taken into

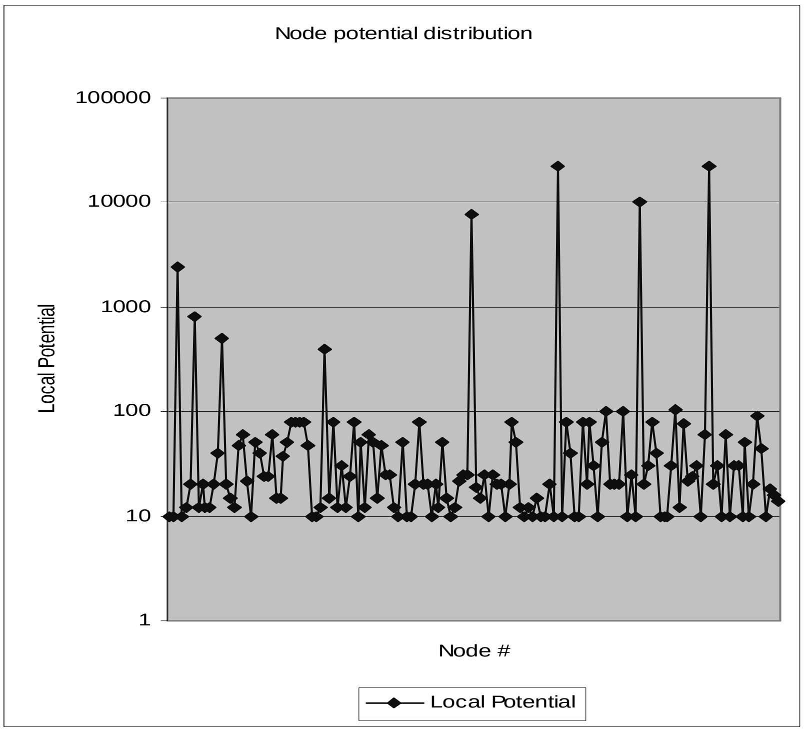

Figure 5.10: Distribution of Local potentials to the nodes in the heterogeneous grid for 100 nodes

consideration in which diverse set of computational resources are present in grid.

Results:The Figure 5.11 shows the Average Message overhead across all protocols and Figure 5.12 shows the Average Dissemination error across all protocols in a 100-nodes heterogeneous grid.

Observation:It is noted in Figure 5.11 that the message overhead of randomized protocol (k=0.5) is high as compared to Neighborhood and PDP protocol. This increase in message complexity is due to the low value of $m = 2.0$ as compared to the standard deviation of the data set which is 2377.918. Since the value of m is 2, the difference between remotely induced potential and local potential will be scaled down lower and thus allowing the low probability to be assigned for the destination nodes who induce the potential of relatively higher magnitude than the local potential. This increases the message overhead and thus

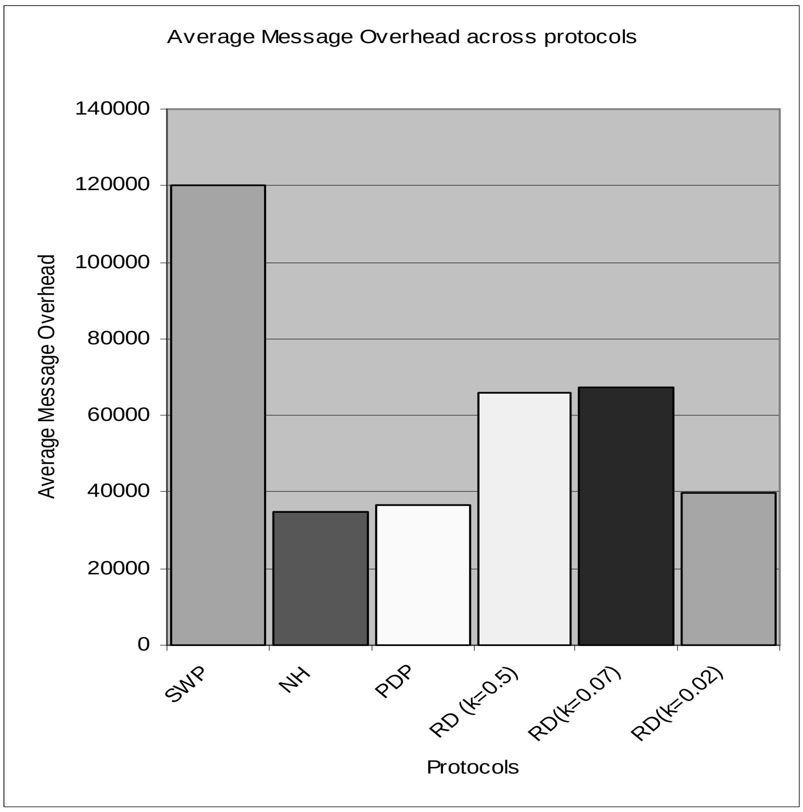

Figure 5.11: Average message overhead across protocols in 100-nodes heterogeneous grid

reduces the error which is noted in Figure 5.12. The performance of the Neighborhood and PDP is also noted to be same in this simulation run. The message overhead of randomized protocol k=0.02 is noted to be same as PDP and Neighborhood protocol and significant reduction in dissemination error is observed in Figure 5.12. This is due to higher value of $m = 50$ as compared to other values of m i.e. 2 and 14. As the value of m increases, the difference between remotely induced potential and local potential will be scaled down (by 50) higher and thus allowing the high probability to be assigned for the destination nodes who induce the potential of relatively higher magnitude than the local potential.

5.2.5 Conclusions

Several conclusions could be drawn from the results obtained by performing the 4 experiments mentioned in the previous subsections. The parameters on which the protocol performance is dependent on are as follows:

1. Connectivity of physical network topology, i.e. grid overlay topology and/or number of bi-components in the topology graph.

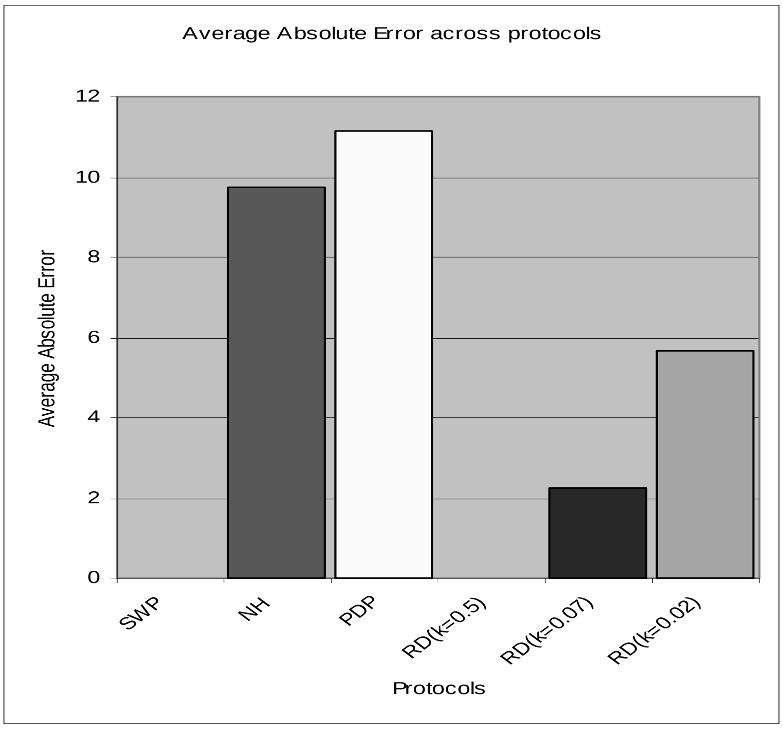

Figure 5.12: Average dissemination error across protocols in 100-nodes heterogeneous grid

2. Type of Grid, i.e. homogeneous or heterogeneous in terms of processing power.
3. Rate of decay of local potential value when disseminated to remote nodes

Clearly, flooding has very low error as compared to the other protocols, at the cost of higher message overhead. The message overhead for Neighborhood protocol is in between PDP and flooding, but the accuracy of neighborhood protocol still remains low as compared to PDP and Randomized protocols. The message overhead of randomized protocol is same as neighborhood protocol in small network size, but the accuracy of randomized protocol is still significant compared to neighborhood protocol and swamping protocol.

Thus, PDP and Randomized protocols are attractive because they exhibit lower overhead compared to Swamping and Neighborhood (relatively small network size) protocols, while maintaining comparable error.

Chapter 6

Conclusion and Future work

This thesis describes the grid resource discovery problem, and outlines the space of solutions that can be used to address it. It is suggested that use of non-uniform dissemination protocols to propagate resource information more accurately to nearby information repositories (as opposed to uniformly across all repositories). This thesis introduces and analyzes two protocols: Prioritized Dissemination Protocol that allows resources to be separated into priority classes, with different forwarding policies implemented for each and Randomized Protocol that further uses the concept of Name-Dropper algorithm [22] that filters the information similar to Prioritized Dissemination Protocol but uses the non-deterministic approach to further improve the performance. The ns-2 simulation of several representative instances of the protocols reveals the characteristics of the tradeoff between message overhead and observed error for different overlay network sizes.

The scalable grid resource discovery is a rich problem domain with many problems. Some of them are as follows:

- More detailed evaluation would include a wider range of overlay backbones and large network size. The analysis of the protocols can be further be extended by considering real grid environment [1].
- The decay rate of the grid potential needs further analysis since it can have a significant impact on the performance of the protocols analyzed
- Another one would be to construct theoretical performance models for data dissemination algorithms that belong to the prioritized dissemination category.
- It is also envisioned that the protocols further learn the resource information variations from the data itself and dynamically adjusts the forwarding probabilities. This

[1]Analyzing in real grid environment requires resource and query traces from a realistic grid environment that is presently deployed. Unfortunately, such information currently is not available

must be supported by intermediate repositories too.

It is believed that such architecture, where data semantics drive the protocol behavior, can enable self-adaptive and fault-tolerant large-scale grid middleware.

Chapter 7

Appendix A

This chapter provides the pseudo-code of algorithms based on "Grid Potential", that are analyzed in this thesis proposed in [32].

7.1 Grid Potential

A node in the Grid has several attributes that can be categorized as rate-based attributes and non rate-based attributes. Examples of rate-based attributes include CPU speed, FLOP rating, sustained memory access rate, and sustained disk access rate. A node in a Grid can be characterized by a vector where each element of the vector is an attribute-value pair.

The Grid potential is based on the computing power or operating rate of a node. Therefore, to characterize a node for deriving the Grid potential only rate-based attributes are considered.

Let $X =< x_0 = a_0, x_1 = a_1, ..x_{n-1} = a_{n-1} >$ Where x_i is a rate-based attribute of the system and a_i its value at a given time.

Let F be a set of functions $f_0, f_1, f_2, .., f_{k-1}$, where f_i operates on the set X to return a scalar value $\lambda_i = f_i(x_0, x_1, .., x_{n-1})$.

Depending on the system, different functions may be defined for it. $\lambda_c = f_c(x_0, x_1, .., x_{n-1})$ may be interpreted as the compute potential.

The storage potential can also be defined like this.

To measure the above parameters, we need a suite of corresponding benchmark programs for each function f_i.

In the suite $\Gamma_i = \{T_0^i, .., T_{n-1}^i\}$, T_j^i is a program specifically designed to evaluate attribute x_j of the node.

One of the benchmark programs might be measuring the rate at which arithmetic operations are being executed.

7.1.1 Node Component Potential

Node component potential p_j^C with respect to attribute x_j is defined as the number of operations performed by node in a second as measured by benchmark program T_j^i

7.1.2 Weighted Node Component Potential

Weighted node capacity is defined as a weighted average of the node component capacities $\{p_0^C, p_1^C, .., p_{n-1}^C\}$ i.e. $p^W = w_0 p_0^C + w_1 p_1^C + .. + w_{n-1} p_{n-1}^C$

7.1.3 Locally Induced Potential

The potential p_i^L induced by a machine i at point of attachment to the grid can be defined as the local grid potential and is defined as $p_i^L = \mu p^W$ where $0 \leq \mu \leq 1$. When the machine is exclusively used for Grid computations, $\mu = 1$ and $0 \leq \mu \leq 1$ otherwise.

7.1.4 Grid Potential

Suppose M machines are attached to a given node j, then the Grid potential at that node is given by $p^G = \max_{i \epsilon [0..M]} \{p_j^L(i)\}$

7.1.5 Remote Potential Drop

The Grid potential induced at the point attachment (node) drops off as we move away from the node along the Grid. This potential drop is dependent on the network characteristics. The Grid potential induced by a machine at a node other than its point of attachment to the Grid is defined as the remote induced Grid potential.
Consider a machine that is attached to the Grid at node i.
Let p_{ij}^R denote the remote induced Grid potential of this machine at node j.

The remote induced Grid potential p_{ij}^R can be considered as the effective processing power of the machine at node j.
Figure 7.1 shows the pseudo-code for computing the potential drop between two nodes on the Grid.
Let s be the source node and d be the destination node i.e., we want to determine locally induced potential at d due to s.
Let p_s be the potential of node s.
Let C be the workload of the benchmark code. The workload is measured in terms of the number of operations.
Let t_s be the execution time of benchmark code on s. Because the node potential is the

execution rate of the benchmark code, $l_s = C/p_s$. The effective processing rate seen at d due to s is given by $C/(t_s + t_c)$. Therefore, the potential drop from s to d is given by $(C \times t_c)/(t_s \times (t_s + t_c))$. The potential drop is dependent on t_s and t_c, where t_c is the communication cost from transferring the benchmark code from s to d. An estimate of t_c may be obtained without actually transferring the benchmark code. Measured values of effective network bandwidth and delay parameters may be used for estimating t_c.

```
calc_potential_drop ($s$,$d$)
//this function returns the Grid potential of node $n$
//for simplicity, it uses a single benchmark code
//let $C$ be the workload of the benchmark code
// the workload is measured in terms of the number of operations
$p_s$ = Grid potential of node $s$
$t_s$ = execution time on node $s$ for the benchmark code
$t_c$ = communication cost of transferring code from $d$ to $s$
$t_s = C/p_s$
return $(C \times t_c)/(t_c \times (t_c + t_s))$
end
```

Table 7.1: Pseudo-code for computing the Grid potential drop. Reproduced from [32]

7.2 Dissemination Algorithms

```
// Universal algorithm
while (true) {
// process incoming messages receive message (X)
{
if (X is meant for local node)
// the neighbor list of nodes may grow if the sender is new
else
{
route the message towards its destination
}
}
if ( currentTime > lastSentTime + n)
{
lastSentTime = currentTime
// get the list of neighboring nodes
for current node get the neighbor list Y
for each node in Y
send a status update message
}
}
```

Table 7.2: Pseudo-code of Universal Awareness Algorithm. Reproduced from [32]

```
// Neighborhood algorithm
while (true)
{
// process incoming messages
receive message (X)
{
if (X is meant for local node)
update the local status information table
// the neighbor list of nodes may grow if the sender is new
}
}
if (currentTime > lastSentTime + n)
{
lastSentTime = currentTime
// get the list of neighboring nodes for current node
get the neighbor list Y for each node in Y
if (dist(node in Y,currentNode) < radius)
send a status update message
}
}
```

Table 7.3: Pseudo-code of Neighborhood Algorithm. Reproduced from [32]

```
// Prioritized algorithm
while (true) {
induced due to remote node in X
update the avg. remote potential at local node
if (X is meant for local node)
update the local status information table
// the neighbor list of nodes may grow if the sender is new
else
{
if (local potential < locally induced remote potential)
route the message towards its destination
}
}
if (currentTime > lastSentTime + n)
{
lastSentTime = currentTime
if (local potential > avg. locally induced remote potential)
for current node get the neighbor list Y
for each node in Y send a status update message
}
}
```

Table 7.4: Pseudo-code of Prioritized Dissemination Algorithm. Reproduced from [32]

```
// Randomized algorithm
// process incoming messages
receive message (X)
{ compute the local potential that is induced due to remote node in X
update the avg. remote potential at local node
if (X is meant for local node)
update the local status information table
// the neighbor list of nodes may grow if the sender is new
else
{ route the message towards its destination
}
}
if (currentTime > lastSentTime + n)
{ lastSentTime = currentTime for current node get the neighbor list Y
for each node in Y compute the sending probability for the destination
// use the probability to decide whether to send status
if (send status)
send a status update message
}
}
```

Table 7.5: Pseudo-code of Randomized Dissemination Algorithm. Reproduced from [32]

References

[1] “Global grid forum,” WebSite, http://www.gridforum.org.

[2] “Globus alliance,” WebSite, http://www.globus.org.

[3] “Sun developer network,” WebSite, http://java.sun.com.

[4] “Tracegraph - network simulator ns trace files analyzer,” WebSite, http://www.geocities.com/tracegraph.

[5] G. Allen, D. Angulo, I. T. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf, “The cactus worm: Experiments with dynamic resource discovery and allocation in a grid environment,” *CoRR*, vol. cs.DC/0108001, 2001.

[6] A. R. Butt, R. Zhang, and Y. C. Hu, “A self-organizing flock of condors,” in *SC ’03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 42.

[7] R. Buyya, D. Abramson, and J. Giddy, “Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid,” 2000.

[8] K. L. Calvert, M. B. Doar, and E. W. Zegura, “Modeling internet topology,” *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, June 1997.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001.

[10] J. H. Cowie, D. M. Nicol, and A. T. Ogielski, “Modeling the global internet,” *Computing in Science and Engg.*, vol. 1, no. 1, pp. 42–50, 1999.

[11] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, “Grid information services for distributed resource sharing,” 2001.

[12] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, “A resource management architecture for metacomputing systems,” *Lecture Notes in Computer Science*, vol. 1459, pp. 62–??, 1998.

[13] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, “Building peer-to-peer systems with Chord, a distributed lookup service,” 2001, pp. 81–86.

[14] P. A. Fishwick, “Simpack: Getting started with simulation programming in c and c++,” in *Winter Simulation Conference*, 1992, pp. 154–162.

[15] I. Foster and C. Kesselman, “Globus: A metacomputing infrastructure toolkit,” *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, Summer 1997.

[16] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, “The physiology of the grid: An open grid services architecture for distributed systems integration,” 2002.

[17] I. Foster, “The grid: A new infrastructure for 21st century science,” *Physics Today*, no. 55, pp. 42–47, 2002.

[18] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., San Francisco–USA, 1999.

[19] I. T. Foster, “The anatomy of the grid: Enabling scalable virtual organizations.” in *CCGRID*, 2001, pp. 6–7.

[20] P. García, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo, “Planetsim: A new overlay network simulation framework.” in *SEM*, 2004, pp. 123–136.

[21] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, “Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead,” in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS ’03)*, 2003.

[22] M. Harchol-Balter, T. Leighton, and D. Lewin, “Resource discovery in distributed networks,” in *PODC ’99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 1999, pp. 229–237.

[23] W. Hoschek, “The web service discovery architecture,” in *SC ’02: Proceedings of the Proceedings of the IEEE/ACM SC2002 Conference*. Washington, DC, USA: IEEE Computer Society, 2002, p. 38.

[24] F. W. Howell and R. McNab, “Simjava: a discrete event simulation package for Java with applications in computer systems modelling,” in *Proceedings of the First International Conference on Web-based Modelling and Simulation*. San Diego, CA: The Society for Computer Simulation, 1998.

[25] A. Iamnitchi and I. Foster, “On fully decentralized resource discovery in grid environments,” in *International Workshop on Grid Computing*. Denver, Colorado: IEEE, 2001.

[26] M. E. Jerrell and W. A. Campione, “The network-enabled optimization system (neos) - a means of solving optimization problems over the internet,” Society for Computational Economics, Tech. Rep. 87, Apr. 2001, available at http://ideas.repec.org/p/sce/scecf1/87.html.

[27] R. M. Jinyang Li, Jeremy Stribling and M. F. Kaashoek, “Accordion: A peer to peer protocol,” WebSite, 2005, http://pdos.csail.mit.edu/ jinyang/pub/.

[28] M. Kaashoek and D. Karger, “Koorde: A simple degree-optimal distributed hash table,” 2003.

[29] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web,” in *ACM Symposium on Theory of Computing*, May 1997, pp. 654–663.

[30] C. Law and K. Siu, “An o(log n) randomized resource discovery algorithm,” 2000.

[31] D. Lu and P. A. Dinda, “Gridg: generating realistic computational grids,” *SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 4, pp. 33–40, 2003.

[32] M. Maheswaran and K. Krauter, “A parameter-based approach to resource discovery in grid computing system,” in *GRID ’00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. London, UK: Springer-Verlag, 2000, pp. 181–190.

[33] P. Maymounkov and D. Mazi&#232;res, “Kademlia: A peer-to-peer information system based on the xor metric,” in *IPTPS ’01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 53–65.

[34] R. McNab, “Simjava: a discrete event simulation library for java,” WebSite, July 1996, http://www.dcs.ed.ac.uk/home/rmcn/simjava.

[35] R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,” in *ACM SIGCOMM 2001*, San Diego, CA, September 2001.

[36] G. P. J. Mrk Jelasity, Alberto Montresor, “Peersim peer-to-peer simulator,” WebSite, March 2004, http://peersim.sourceforge.net/.

[37] J. S. Plank, H. Casanova, J. J. Dongarra, and T. Moore, “Netsolve: An environment for deploying fault-tolerant computing,” in *FastAbstracts Session, FTCS-28: 28th International Symposium on Fault-tolerant Computing*, Munich, June 1998.

[38] O. F. Rana, D. Bunford-Jones, K. A. Hawick, D. W. Walker, M. Addis, and M. Surridge, “Resource discovery for dynamic clusters in computational grids,” in *IPDPS ’01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*. Washington, DC, USA: IEEE Computer Society, 2001, p. 82.

[39] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content addressable network,” Berkeley, CA, Tech. Rep. TR-00-010, 2000.

[40] M. Ripeanu, “Peer-to-peer architecture case study: Gnutella network,” 2001.

[41] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” *Lecture Notes in Computer Science*, vol. 2218, pp. 329+, 2001.

[42] J. L. R. M. J. S. Thomer M. Gil, Frans Kaashoek, “p2psim: a simulator for peer-to-peer protocols,” WebSite, April 2005, http://pdos.csail.mit.edu/p2psim/.

[43] A. E. Walsh, Ed., *Uddi, Soap, and Wsdl: The Web Services Specification Reference Book*. Prentice Hall Professional Technical Reference, 2002.

[44] E. Zegura and K. Calvert, “Gt internetwork topology models (gt-itm),” WebSite, http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.

[45] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in *IEEE Infocom*, vol. 2. San Francisco, CA: IEEE, March 1996, pp. 594–602.

[46] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An infrastructure for fault-tolerant wide-area location and routing,” UC Berkeley, Tech. Rep. UCB/CSD-01-1141, Apr. 2001.