# Efficient Information Dissemination in Wide Area Heterogeneous Overlay Networks

A Thesis
Presented to
The Academic Faculty

by

**Jianjun Zhang**

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
August 2006

# EFFICIENT INFORMATION DISSEMINATION IN WIDE AREA HETEROGENEOUS OVERLAY NETWORKS

Approved by:


Dr. Ling Liu, Advisor
College of Computing
*Georgia Institute of Technology*

Dr. Constantine Dovrolis
College of Computing
*Georgia Institute of Technology*

Dr. Calton Pu, Co-Advisor
College of Computing
*Georgia Institute of Technology*

Dr. Brian Cooper
College of Computing
*Georgia Institute of Technology*

Dr. Vaidy Sunderam
Department of Mathematics and Computer Science
*Emory University*


Date Approved: July 5, 2006

*To my parents and my wife Han.*

# ACKNOWLEDGEMENTS

This thesis would not have been possible without the generous help and encouragement of many individuals. I would like to express my gratitude to everyone who has contributed to the process leading to my dissertation. Here I would like to mention a few of these people.

First and foremost, I want to thank my advisor Prof. Ling Liu for her tremendous help and guidance in both my Ph.D. study and career development. She provided me the invaluable flexibility and freedom in deciding and developing my research works as well as my career plan. To me, she is not only an acute and knowledgeable academic advisor, but also a supportive and resourceful career mentor. I am indebted for the countless hours she has spent on perfecting my work and the generous advice she has given me on research, career, and life related matters. She has set a perfect example of being a successful researcher and an encouraging adviser for me. Her example will continue to light my way through my future career.

I would like to give my hearty thanks to my co-advisor Prof. Calton Pu for his timely and bountiful advice and generous support. He set an example that I can follow through my future career: to become an insightful scholar and a successful leader as he is. I would also like to thank every member of DiSL research group for providing a friendly and dynamic working environment and for engaging in productive research discussions with me, that have helped in continuously improving and polishing my work.

I would like to thank my committee members Prof. Brian Cooper, Prof. Constantine Dovrolis, and Prof. Vaidy Sunderam for being very supportive of my research and for their constructive critiques of my work that have greatly contributed to this thesis. I would also like to thank Prof. Lakshmish Ramaswamy for his suggestions and help on polishing this thesis.

I would like to dedicate this work to my father, mother, my wife Han Zheng, and my big family for their love, understanding, and gratuitous support for me.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Past a few years witnessed the continuous growth of multi-party group communication applications such as multiplayer online games, online community based advertising, real-time conferencing, and instant messaging. The implementations of these applications usually involve the dissemination of text or multimedia contents among multiple participants. Peer-to-Peer (P2P) overlay networks present a promising paradigm for harnessing widely distributed, loosely coupled, and inherently unreliable peers for supporting distributed applications. Information dissemination in such an environment could be implemented by organizing end-hosts into a spanning tree interconnected by IP unicast links and using end-hosts to forward the communication payload along the edges of the spanning tree. A key challenge in building information dissemination overlay network is the system efficiency. We argue that any wide area information dissemination solution should ensure that the end-to-end communication latency and efficiency in the overlay network be comparable to that of the IP multicast systems. Furthermore, an information dissemination overlay network should provide a general and scalable communication substrate for various distributed multi-party group communication applications.

In this dissertation research we study and address the unique challenges involved in information sharing and dissemination for large-scale multi-party group communication applications. We focus on system architectures and various techniques for providing efficient and scalable information dissemination services in distributed P2P environments. Our solutions are developed by targeting at utilizing three representative P2P overlay networks: a structured P2P network based on consistent hashing techniques, an unstructured Gnutella-like P2P network, and a P2P service network where end-system nodes are organized based on their geographical locations and geographical proximity. We have made three unique contributions to the general field of large-scale information sharing and dissemination. First, we propose a landmark-based peer clustering technique to grouping end-system nodes by

their network proximity, and a communication management technique to address load balancing and reliability of group communication applications in structured P2P networks. Second, we develop a utility-based P2P group communication service middleware, consisting of a utility-based topology management protocol and a utility-aware P2P routing protocol, for providing scalable and efficient group communication services in unstructured P2P overlay networks of heterogeneous peers. Third, we propose a service network management protocol that is aware of the geographical location of end-system nodes and a set of geo-proximity based routing and adaptation techniques, aiming at building decentralized information dissemination service networks to support location-based applications and services. Our experiments show that our solutions can offer significant improvements over the existing approaches in term of communication efficiency, system scalability, and system load balancing.

Although different overlay networks require different system designs for building scalable and efficient information dissemination services, we have employed two common design philosophies: i) exploiting end-system heterogeneity and ii) utilizing proximity information of end-system nodes to localize most of the communication traffic, and using randomized shortcuts to accelerate long-distant communications. We have demonstrated our design philosophies and the performance improvements in all three types of P2P overlay networks. By assigning more workloads to more powerful peers, we can greatly increase the system scalability and reduce the variation of workload distribution. By clustering end-system nodes based on their IP-network proximity or their geographical proximity, and utilizing randomized shortcuts, we can reduce the end-to-end communication latency, balance routing workloads against service request hot spots across the overlay network, and significantly enhance the scalability and efficiency of distributed information dissemination and decentralized multi-party group communication.

# CHAPTER I

# INTRODUCTION

With rapid growth of wireless communication technology and increasing popularity of hand-held devices, we witness the continuous escalation of multi-party group communication applications, such as multiplayer online games, online community based advertising, real-time conferencing [4], and instant messaging [3]. The implementations of these applications usually involve the dissemination of text or multimedia contents among multiple participants. The participants could serve as the receiver or sender of the contents. They could be static or mobile nodes, using wired or wireless connections to access the network.

There is an increasing demand for effective mechanisms for low-cost, scalable, and efficient information dissemination to a large number of end users. Several multi-party group communication solutions have been proposed in the literature [4, 3, 6]. They are either centralized client-server systems or decentralized Peer-to-Peer (P2P) systems. However, none of them, to our knowledge, has fully addressed all the technical challenges of large scale group communication applications.

Most of existing centralized group communication systems achieve scalability through computer cluster-based server farms [4, 3], which are exclusively owned by service providers and require significant administrative investment. Subscribers usually have to pay a premium for the services [4] and are often limited to the service functions defined by the specific service provider.

The success of Skype [6] has shown both the opportunity and the feasibility of using P2P overlay network as an economical alternative service infrastructure for providing scalable wide-area group communication services. In Skype, widely distributed personal computers are interconnected by an unstructured P2P overlay network. By exploiting the high flexibility and low maintenance cost of such an open system architecture, Skype enables value-added services like Internet-to-PSDN calls at a fraction of the price of traditional

telephony services. However, the overlay network in Skype is typically used only for service lookup and control signaling. Under the multi-party conference settings, the voice communication payloads are directly forwarded to and relayed by a single node through its IP unicast links [13]. As a result, the maximum number of participants allowed in each conference session is limited, e.g., less than 6 in Skype.

This dissertation is focused on research issues and challenges in efficient information dissemination in wide area heterogeneous overlay networks, especially on information dissemination to large scale user groups. We first provide some background information on information dissemination to large scale user groups, motivating group communication applications. Then we describe the properties of the heterogeneous overlay networks and present the research challenges for building scalable and efficient information dissemination applications using an overlay network of heterogeneous nodes.

## *1.1 Information Dissemination to Large Scale User Group*

Compared to traditional point-to-point communication model, information dissemination model for large scale user groups has three distinguishing characteristics.

- one or many information sources supply the information to be disseminated. The number of information sources may be uncertain at the time when an application instance is launched.

- a number of receivers act as the consumers of information in the user groups. Depending on the contents of the information disseminated and the properties of the network connections those receivers have, they may have different quality of service requirements and different tolerance to network delay or failures.

- a spanning tree that carries the communication payloads. Propagated information may be replicated or processed in the spanning tree before they reach the receivers. Traditional client/server architecture can be viewed as a special spanning tree of height 1, in which information replication is at the root of the spanning tree. Such an implementation requires tremendous processing power and network bandwidth at

the server side to support a large and possibly growing number of participants.

## *1.2 Example Applications*

Due to the capability of reaching a large number of receivers, information dissemination system supporting large scale user groups can be used to implement a wide spectrum of applications. Here we describe three application examples from different application domains:

i) End System Multicast

ii) Multi-participant Conferencing

iii) Location-based Publish/Subscribe System.

### 1.2.1 End System Multicast

End System Multicast could be used to implement applications that require one-to-many communication. Examples of such application include online broadcast of text or multimedia information, which enables a large number of receivers to watch or listen to live shows, contents deliver networks for updating web information, and software update services that push new update patches to end users. Existing solutions includes SCRIBE [17], ScatterCast [18], Navada [20], NICE [12], and Bayeus [67], just to name a few. They were proposed to replace IP multicast [22, 23] systems that are not widely accepted and deployed today.

### 1.2.2 Multi-Participant Conference

This application scenario is more appealing to business users. It is the implementation of many-to-many communication paradigm. Participants of the conference could hear or see the other participants by receiving the audio or video contents of them, and make themselves seen and heard by uploading their own audio and video signals. Existing solutions include centralized ones like Google Talk [3] and Microsoft Livemeeting [4]. Decentralized solutions include P2P VoIP systems such as Skype [6] and PeerCast [62], which we will present in Chapter 3 of this dissertation.

### 1.2.3 Mobile Location-based Publish/Subscribe System

With rapid advances in wireless and mobile communication technologies, there is a growing demand for providing efficient and scalable publisher-subscriber solutions for mobile location based applications. An example subscription that could be supported by mobile publish/subscribe applications is "please inform me of the traffic around Exit 89 on I-85 in the next 30 minutes". Sensors and other information monitors near the requested location capture the events of interests (e.g., the traffic around Exit 89 on I-85 in this example). Events once detected are delivered to all subscribers by matching the description or the content of the events to the subscription(s) submitted by the subscribers. Due to the nature of end-user mobility, both information providers and receivers may constantly change their locations. Thus system-level facilities are required for efficiently routing, scalable processing, and reliable and timely delivery of events in such a mobile publish/subscribe network.

## *1.3 Challenges and Issues*

We have described three classes of group communication applications that use overlay networks as the communication and computing platforms to support large scale information disseminations. Peer-to-Peer (P2P) networks demonstrate a promising paradigm for providing distributed interactive applications and distributed information sharing services by harnessing widely distributed, loosely coupled, and inherently unreliable computer nodes (peers) at the edge of the Internet. Two typed of P2P networks exist, i.e., structured P2P networks and unstructured P2P networks.

- Structured P2P networks [45, 49, 64, 52] regulate the network topologies through Distributed Hash Tables (DHT) and provide efficient inter-peer communication in a bounded number of hops. Peers in the network are usually assigned with randomly generated identifiers from a hashing key space. Objects in the network are assigned with identifiers from the same key space, and are mapped onto corresponding peers. By progressively mapping identifiers to the locally maintained routing information on peers, searching messages are processed and forwarded in a deterministic manner.

- In contrast, unstructured P2P networks represented by Gnutella [58] and Kazza [5] have low maintenance cost against network dynamics such as peer joining, failure, and departure. Peers in an unstructured P2P network are usually connected to one another in an ad-hoc manner. Objects are stored locally on peers. Each peer usually maintains only the local neighbor information. Searching in unstructured P2P network is commonly implemented as a nondeterministic yet controlled flooding search.

### 1.3.1 Common Challenges

In this dissertation research we focus on addressing three challenges that are common to the P2P overlay networks in the context of building efficient, scalable, and reliable information dissemination applications. These challenges are described as follows:

- **Efficiency** There exists mismatching between the P2P overlay network topology and the underlying physical network topology, which has been well studied in the literature [45, 49, 60]. In most generic P2P networks, the indexing techniques and the routing schemes are completely independent and oblivious to the underlying network structure. Hence, communication in these networks is likely to be very inefficient in terms of the physical network route traversed by individual message. When P2P overlay networks are used as the communication platforms for information dissemination, communication payloads may be sent multiple times over the same physical network link. Such mismatching will impose high communication overheads such as redundant network traffic and communication delay.

- **Load Balance** The second challenge is to maintain load balance among the heterogeneous peers. One of the main causes of load imbalance is the heterogeneity in the resource availabilities at various nodes. Ideally the workload assigned to each peer should be proportional to its capabilities. This would prevent nodes from becoming bottlenecks, thereby improving the efficiency and dependability of the system. Therefore, it is essential to augment the communication management schemes with techniques that distribute the communication workload among nodes in the system according to their resource availabilities.

- **Reliability** End-system nodes are dynamic and are more prone to failures. It is widely recognized that large-scale distributed systems like P2P network are confronted with high turnover rate [50], with nodes entering and departing the system at arbitrary points in time. Ensuring high communication service availabilities in such dynamic systems is crucial for the success of large scale information dissemination.

We are interested in exploring scalable, efficient, and reliable system-level facilities and methods for building large scale group communication services and disseminating information using P2P overlay networks. We are interested in examining the common challenges to both structured and unstructured P2P networks and the specific challenges posted by the different design methods used in P2P topology formation and routing protocol designs. In addition, we are interested in geographical location-based P2P networks − a type of specialized structured P2P networks. In these P2P overlay networks, the peer identifier space is mapped to a geographical plane in which peers physically reside. Each peer is assigned with a rectangular area of the geographical plane. Objects in the overlay are identified by their geographical coordinates. Each object is mapped to a peer whose rectangular area covers its geographical coordinate. Routing and searching in a location-based P2P network is carried out in a deterministic manner by passing message from peer to peer. A routing request is usually tagged with the coordinate of the target. At each routing step, a request is forwarded from a peer to one of its neighbors that is the closest to the coordinate of the target point.

Now we discuss the specific research challenges in each of these three types of P2P networks.

### 1.3.2 Specific Challenge in Structured P2P Network

The peer identifiers in structured P2P overlay networks are usually generated using a hashing function, following a random distribution. The efficiency and the deterministic manner of routing in structured P2P overlay networks relies on the randomness of the identifier space for guarantee. If we modify the distribution of identifier space to match it to the topology of underlying physical networks, we may destroy the randomness of identifier

space, and cause suboptimal routing performance. Thus an interesting challenge in this type of overlay networks is to find a scheme that can preserve the randomness of identifier space, and improve the routing efficiency by matching the topology of an overlay network to the topology of its underlying IP network.

### 1.3.3 Specific Challenge in Unstructured P2P Network

Metrics such as network proximity and node capacity have been recognized in the literature as critical parameters for optimizing P2P communication in wide area networks. Although each of these metrics has been studied with one specific system optimization objective in mind, to our best knowledge, very few research works have addressed the issue of combining these metrics to achieve the multiple system optimization objectives demanded by large-scale wide-area group communication applications. Important research questions include how to combine the network proximity metric and node capacity metric to maximize the efficiency of group communication applications while providing fast P2P lookup. In the situations where these two metrics cannot be combined in a straightforward manner, how can we utilize the benefits of one metric with the minimal downgrade of the benefits of the other metric?

### 1.3.4 Specific Challenge in Mobile and Location-base Service Network

We identify two features that are critical yet not fully supported by current P2P solutions. First, the P2P network should provide efficient support for location-based routing. Information required by mobile users is usually tagged with their location information. Typical queries like "where is the nearest gas station?" and "show me the three nearest available parking lots" require the processing of online data by the location of end-users. To support such services using P2P networks, current P2P routing protocols need to be extended so that both queries and answers could be efficiently delivered over the unicast links among peers.

Second, the location-based information is often not evenly distributed and usually exhibits certain spatial and temporal clustering patterns. For example, during a sport event, the parking lots around the stadium are usually full, whereas in other days with no sport

events, those parking lots are sparsely used. To support location-based queries using a P2P overlay network of heterogeneous end-systems, the P2P network should be able to handle such hot spots and unbalanced workload distribution gracefully, minimizing the possible service interruptions caused by sudden load peaks on some portion of the nodes in the overlay.

## *1.4 Contribution of the Thesis*

This dissertation research aims at developing system-level architectures and techniques to support information dissemination to large scale user groups by using P2P overlays, including structured and unstructured P2P overlays, and P2P overlays constructed using geographical location information. The main contributions of this dissertation research can be summarized into three parts:

1. We design and develop a self-configuring, efficient and reliable end-system multicast system called *Cascade*. Three unique features distinguish Cascade from existing approaches to application-level multicast systems.

    - First, with the aim of exploiting network proximity of end-system nodes for efficient multicast subscription management and fast information dissemination, we propose a novel Internet-landmark signature technique to cluster end system nodes in the overlay network by their physical network proximity.
    - Second, we propose a capacity aware overlay construction technique to balance the multicast workload among heterogeneous end-system nodes.
    - Third, we develop a dynamic passive replication scheme to provide reliable end system multicast services in an inherently dynamic environment of unreliable peers. We also present a set of experiments, showing the feasibility and the effectiveness of the proposed mechanisms and techniques.

2. We propose and develop *PeerCast*, a utility-based P2P group communication middleware for providing scalable and efficient group communication services in an unstructured P2P overlay network of heterogeneous peers. We observe that nodes with

different capacities tend to have different preferences over the network proximity and the node capacity metrics. Our main contribution is to carefully combine these two metrics in our P2P overlay management and communication group management protocols. To our knowledge, PeerCast is the first P2P group communication system that is built using a low-diameter unstructured P2P network, which follows a power-law topology. Unique features of PeerCast include:

- A service announcement mechanism that selectively propagates the service information to peers in the overlay network. By cutting out the paths that will not likely be used in a group communication spanning tree, this mechanism reduces the messaging network traffic by 27% to 56% compared to the popular advertising scheme used in [18], without affecting the performance of group communication applications.
- A fast service lookup mechanism that enables a participant to discover the services of interest with fewer probing messages. Our service announcement mechanism pushes the group communication information closer to the participants, such that they can locate the service of interest within their overlay network neighborhood much faster, and using fewer searching messages.
- A communication group management mechanism that constructs efficient spanning trees for wide-area group communications. Our experiments indicate that in most of the cases, the end-to-end communication latency between any two peers in the spanning tree is within 3 times of the IP unicast latency. By matching the communication workloads of each peer to its capability, we are able to reduce the overloading in the overlay network by one to two orders of magnitudes.

3. We design and develop a scalable service network, called *GeoGrid*, for supporting efficient location-based information dissemination applications. Our design exhibits three unique features that distinguish it from existing solutions.

   - First, GeoGrid design supports geographical location-based topology formation

and P2P routing with effective load-balance and fault-tolerance as its design objectives.

- Second, GeoGrid system can provide scalable and efficient routing services, by adding randomized shortcuts that can greatly improve the routing efficiency. In a GeoGrid system with $N$ peers, we achieve an average $O(\log N)$ end-to-end routing delay in terms of routing hops.
- Third but not the least, GeoGrid is equipped with a set of novel features that can efficiently utilize the heterogeneous capacities of end-systems and dynamically balance both the routing workload and information dissemination workload in response to the dynamic workload distribution. Our experimental study demonstrates that the GeoGrid can effectively reduce the workload imbalance by an order of magnitude.

Although different overlay networks require different system designs for building scalable and efficient information dissemination services, the research works presented in this dissertation share and employ three common design philosophies:

- exploiting end-system heterogeneity.
- utilizing proximity information of end-system nodes to localize the communication traffic.
- using randomized shortcuts to accelerate long-distant communication.

In this thesis, we demonstrate our design philosophies and their impacts on the performance improvements in all three types of P2P overlay networks. Concretely, by statically and dynamically assigning more workloads to more powerful peers, we can greatly increase the system scalability and reduce the variation of workload distribution. By clustering end-system nodes based on their IP-network proximity or their geographical proximity, and by utilizing randomized shortcuts, we can reduce the end-to-end communication latency, balance peer workloads against service request hot spots across the overlay network, and

significantly enhance the scalability and efficiency of distributed information dissemination and decentralized multi-party group communication.

In summary, this is, to our knowledge, the first dissertation work that implements the above three design principles in an unstructured P2P overlay network for large scale group communication. It is also the first work proposed to incorporate the above three design philosophies in a location-based P2P overlay network to support various information dissemination applications.

## *1.5 Organization of the Thesis*

The rest of this thesis is organized as a series of chapters, each one dedicated to a specific topic within the context of structured P2P, unstructured P2P, and location-based P2P overlay networks. In each of these chapters, background information and system models are given before the core technical content is described. The specific contributions are given in the introduction part of each chapter, whereas the related work in the literature is reported at the end of each chapter. Concretely, this thesis is composed of the following chapters.

Chapter 2 presents the Cascade end-system multicast system which is based on a structured P2P overlay. Three major components of the Cascade, namely the network proximity based end-system clustering, the virtual-node based load-balancing scheme, and a dynamic replication scheme are presented. Several experimental results are presented to study the system and routing efficiency, load balance, and reliability of the Cascade system.

Chapter 3 presents the PeerCast group communication management system which is based on an unstructured P2P overlay. Three major components of the PeerCast system, namely a utility metric combining the evaluations on both network proximity and node capacity metrics, the bootstrapping and maintenance protocol for building unstructured P2P network following power-law distribution, and an efficient service lookup and communication group management protocol are presented. Several experimental results are presented to study the system and routing efficiency, and load balance of the PeerCast system.

Chapter 4 presents the GeoGrid location-based service network. Three major components of the GeoGrid system, namely a location-based overlay management protocol, a randomized shortcuts management protocol for fast routing, and a dynamic service load balance protocol are presented. Several experimental results are presented to study the routing efficiency, and the system and routing load balance of the GeoGrid system.

Chapter 5 discusses some open issues and concludes the thesis.

# CHAPTER II

# INFORMATION DISSEMINATION USING END SYSTEM MULTICAST

## *2.1 Introduction*

In this chapter, we study the problem of using End System Multicast (ESM) for large scale information dissemination. In recent years end system multicasting (application-level multicasting) has emerged as a practical alternative to IP level multicasting for disseminating information to large sets of receivers [12, 17, 18, 20, 35, 44, 67, 46]. However, supporting end system multicast in a dynamic Internet-scale environment poses a number of challenges. First, an ESM system usually replicates data on end-hosts and propagates them through multi-hop IP unicast links. A critical challenge for ESM system design is to achieve high efficiency and minimize multicast latency experienced by the end-hosts. Second, end-hosts from a wide-area network tend to vary widely in terms of their computing capacities, access network bandwidths, and willingness and ability to share their resources. Such heterogeneity manifests itself as the variations in the amount of workloads the different nodes can handle. Therefore, there is a need for an end system multicast protocol that can organize end-hosts into efficient multicast overlays, and effectively balance multicast workloads on them. Third, it is widely recognized that large-scale distributed systems like Peer-to-Peer(P2P) network are confronted with high turnover rate [50], with nodes entering and departing the system at arbitrary points in time. Ensuring high multicast service availabilities in such dynamic systems is crucial for the success of end system multicast.

Research in this area has mostly focused on mitigating the first challenge [12, 17, 18, 20, 35, 44, 67, 46]. In contrast, the second and third challenges have received very little research attention. We believe that these distinct challenges are in fact very related. Unfortunately, none to our best knowledge has comprehensively addressed these problems. Further, even

the schemes proposed to counter the efficiency challenge suffer from significant limitations.

In this chapter, we present *Cascade* – an efficient, self-configuring, and reliable end system multicast service, which is built on top of an overlay network of loosely coupled and possibly unreliable end-system nodes. Our ESM service can enable group communication capabilities in generic P2P networks, thereby providing a platform for advanced applications such as audio video conferencing, event and content distribution, and multi-user games.

Cascade uses a structured P2P network protocol to organize end-hosts into an overlay network, and builds ESM applications using the P2P network as the communication substrate. While a few existing systems addressed similar problems, our approach has three unique features.

First, we develop a decentralized mechanism to effectively cluster end-hosts by their physical network proximity in the Cascade P2P network. Our novel multicast group management protocol utilizes these clusters to build efficient multicast trees, so that latencies and overheads of the multicast information dissemination are minimal. Second, we propose a capacity-aware overlay construction technique to balance the multicast load among heterogeneous peers. This scheme can effectively distribute the workload among end-hosts. To the best of our knowledge, Cascade is the first ESM system that takes end-system heterogeneity into account. Further, our scheme also encourages peers to share more resources by according better services to the peers contributing more resources. Third, we develop a dynamic passive replication scheme in order to provide reliable end system multicast service in an environment of inherently unreliable peers. This chapter reports a set of experiments that we have performed to evaluate the proposed techniques. The results indicate that the Cascade system is highly efficient, and it exhibits very good load-balancing and reliability characteristics.

## *2.2 Fundamental Concepts and Notations*

In this section, we discuss the fundamental concepts of multicasting and formalize the end system multicasting problem.

**Definition 1** *A **Physical Network** is a directed and connected graph* $G_{physical} = (V_{core} \cup$

$V_{end}, E_{core} \cup E_{end})$, *where* $G_{core} = (V_{core}, E_{core})$ *is a subgraph such that* $E_{core} \subseteq V_{core} \times V_{core}$. $G_{end} = (V_{core} \cup V_{end}, E_{end})$, $E_{end} \subseteq V_{end} \times V_{core} \cup V_{core} \times V_{end}$ *is another subgraph satisfying the condition that* $\forall v_{end_i} \in V_{end}, \exists v_{core_j} \in V_{core}$ *such that* $(v_{end_i}, v_{core_j}) \in E_{end} \wedge (v_{core_j}, v_{end_i}) \in E_{end}$.

By Definition 1, we model an ESM system as a connected, directed, and weighted graph $G_{physical}$. This graph consists of a core network $G_{core}$ and a set of end-hosts $V_{end}$. The core network $G_{core}$ models the IP network and is a connected, directed, and weighted subgraph of $G_{physical}$. $G_{core}$ consists a set of nodes $V_{core}$, which are connected by a set of links $E_{core}$. $V_{core}$ models network devices such as IP network routers. The end-hosts and their access network links are represented by $G_{end}$, another subgraph of $G_{physical}$. For each end-host $v_{end}$, there always exists an access network link $e_{end}$ that connects $v_{end}$ to at least one node $v_{core}$ in the IP network. Each edge $e \in E_{core} \cup E_{end}$ is a directed edge and can be associated with different properties, depending on the application built over the physical network. In our case , we assign each edge $e$ a weight $w_e$ and a length $l_e$. We use $w_e$ to model the IP packet routing weight and use $l_e$ to model the latency of link $e$.

A *Route* between two end-hosts $v_0$ and $v_i$ is defined as a list of edges $((v_0, v_1), (v_1, v_2), \cdots, (v_{i-1}, v_i))$, where $v_0 \in V_{end}$, $v_i \in V_{end}$, $\forall_{1 \leq j \leq i-1}\ v_j \in V_{core}$, $(v_0, v_1) \in E_{end}$, $(v_{i-1}, v_i) \in E_{end}$, and $\forall_{1 \leq j \leq i-2}\ (v_j, v_{j+1}) \in E_{core}$. The *Route Weight* of route $((v_0, v_1), (v_1, v_2), \cdots, (v_{i-1}, v_i))$ is the summation of the weight of each edge on this path, i.e. $\Sigma_{j=0}^{i-1} w_{(v_j, v_{j+1})}$. And the *Path Length* of this route is equal to $\Sigma_{j=0}^{i-1} l_{(v_j, v_{j+1})}$. A route is one of the passible pathes that IP network packets can travel between two end-hosts. The *Path Length* is the accumulative latency of such a route.

We model IP unicast paths as:

**Definition 2** *An* ***IP Unicast Path*** *between two end-hosts* $v_i$ *and* $v_j$ *is a route between that two end-hosts with the minimum route weight. We denoted it as* $UPath_{<v_i, v_j>}$ *and refer to the path length as its Unicast Latency.*

We use an example to illustrate our network model. In Figure 1, $V_{core} = \{R1, R2\}$, $V_{end} = \{A, B, C, D\}$, $E_{core} = \{< R1, R2 >, < R2, R1 >\}$ and $E_{end} = \{< A, R1 >, <$

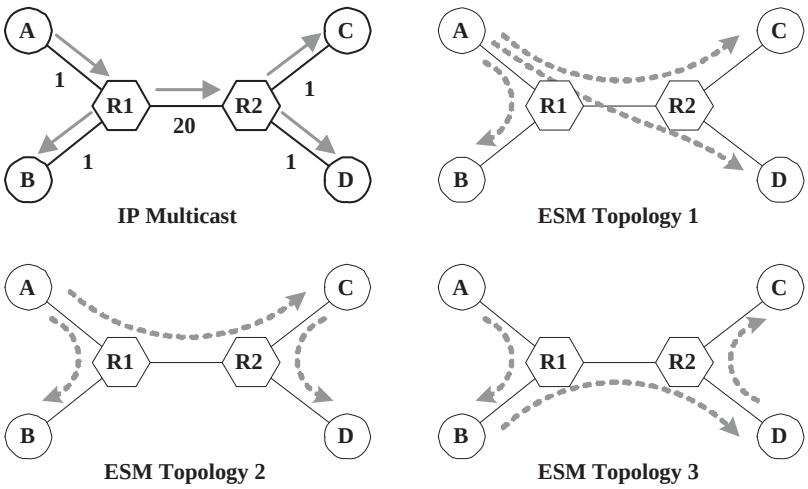


**Figure 1:** End System Multicast Example

$R1, A >, < B, R1 >, < R1, B >, < C, R2 >, < R2, C >, < D, R2 >, < R2, D >\}$. We assume the edges between any pair of nodes are symmetric, and have equal weight and length value, denoted by the number over each edge in the figure. The unicast path between node $A$ and $D$, $UPath_{A,D}$, is $((A, R1), (R1, R2), (R2, D))$, and the unicast latency of this path is 22.

From so on, we will assume that the network links in our graph are symmetrical. Although in many cases, network links may present different properties in different directions, our result will still be valid without this assumption.

An IP multicast tree built over a physical network is an acyclic and connected subgraph of $G_{physical}$ that connects all the end-hosts participate the multicast group. We define an IP multicast tree as:

**Definition 3** *An **IP Multicast Tree** over graph $G_{physical}$ is a directed shortest distance spanning tree in graph $G_{physical}$, with the IP multicast source end-host $v_{root} \in V_{end}$ as the root, all the subscriber end-hosts $V_{subscriber} \subseteq V_{end}$ as the leaves of the tree, and nodes belong to $V_{core}$ as the intermediate nodes in the tree. We denoted it as $T_{IPM}(G_{physical}, v_{root}, V_{subscriber})$*

To measure the performance of ESM overlay networks against IP multicast systems, we assume that the IP multicast systems have optimal multicast trees, i.e., the path between the multicast root node and any receiver node is the same as the IP unicast path between them.

The *IP Multicast Delay* $L_{IP<root,end>}$ between the IP multicast root $v_{root}$ and an end-host $v_{end}$ in tree $T_{IPM}(G_{physical}, v_{root}, V_{receiver})$ is the unicast latency of path $UPath_{<v_{root},v_{end}>}$. For example, in Figure 1, the $L_{IP<A,D>} = 22$.

An ESM overlay network can be conceptualized as a virtual network with all the message forwarding and multicast group management functionalities being pushed to the end-hosts. The link between any two adjacent end-hosts in an ESM overlay network is actually the unicast path between them. We define a complete directed graph $G_{overlay} = (V'_{end}, E)$ based on graph $G_{physical}$, to model ESM overlay networks. We refer to this graph as an *Overlay Graph*.

**Definition 4** *The* ***Overlay Graph*** *of a physical network* $G_{physical} = (V_{core} \cup V_{end}, E_{core} \cup E_{end})$ *is a complete directed graph denoted by* $G_{overlay} = (V'_{end}, E)$*, where* $V'_{end} = V_{end}$ *and* $E = V'_{end} \times V'_{end}$*.*

Each edge $e(i, j) \in E$ of $G_{overlay}$ represents the unicast path between end-hosts $i$ and $j$ in the graph $G_{physical}$, i.e., $UPath_{<i,j>}$. We define the *latency* of edge $e(i, j)$ as $L_{e(i,j)}$, which is equal to the path length of the unicast path $UPath_{<i,j>}$ in graph $G_{physical}$.

An ESM overlay network that delivers multicast payloads from a multicast root end-host $v_{root}$ to a set of subscriber $V_{subscriber}$ is a spanning tree over the overlay graph $G_{overlay}$. We model it as:

**Definition 5** *An* ***End System Multicast Tree*** *over an Overlay Graph* $G_{overlay}$ *is a directed spanning tree defined over* $G_{overlay}$*, with the multicast source end-host* $v_{root}$ *as the root and all the subscribers* $V_{subscriber}$ *being connected in the tree. We denote such a tree as* $T_{ESM}(G_{overlay}, v_{root}, V_{subscriber})$*.*

IP multicast is by nature more efficient than end system multicast. In IP multicast, IP packets are replicated at routers within the physical network. Each multicast message is injected into the physical network only once before it is delivered along the IP multicast tree to the subscriber end-hosts. Whereas in the case of ESM overlay networks, messages of the same contents are replicated on end-hosts and may be injected into the physical network more than once.

The ESM topology 2 of Figure 1 gives such an example. There will be duplicated data packets traveling through links $< A, R1 >$ and $< R2, C >$ because the message replication functions are now on end-hosts $A$ and $C$. In the IP multicast case, the IP packets will be replicated only at router $R1$ and $R2$.

The key concerns in designing an ESM protocol is its efficiency. For each set of subscriber $V_{subscriber}$, a multicast root $v_{end}$, and its corresponding ESM overlay network $G_{overlay}$, we define several performance metrics to measure the ESM tree $T_{ESM}(G_{overlay}, v_{end}, V_{subscriber})$ against the IP multicast tree $T_{IPM}(G_{physical}, v_{end}, V_{subscriber})$. We say an ESM multicast tree is *equivalent* to an IP multicast tree when they have the same multicast root end-host and the same subscriber end-hosts, and are built over the same physical network.

- The *End System Multicast Delay* experienced by an end-host $v_{end}$ is the summation of the unicast latency of edges between the root $v_{root}$ and $v_{end}$ in the end system multicast tree $T_{ESM}(G_{overlay}, v_{end}, V_{receiver})$.

- The *Relative Delay Penalty* of an ESM overlay network is the ratio between the average end system multicast delay of all the subscribers in the ESM tree and the average IP multicast delay of all the subscribers in the equivalent IP multicast tree.

We define link stress as a performance metrics to measure the amount of extra network traffic caused by ESM overlay networks, and to compare it to the equivalent IP multicast systems.

- The *Link Stress* is the ratio between the total number of IP packets imposed by an ESM overlay network on the physical network links and the total number of IP packets imposed by the equivalent IP multicast system, to deliver the same amount of multicast information.

Using ESM topology 2 in Figure 1 as an example. The end system multicast delay of path $A$ to $D$ is 24 and the IP multicast delay between $A$ and $D$ is 22.The relative delay penalty is $(22 + 2 + 24)/(22 + 22 + 2) = 48/46$. On the other hand, the link stress is 7/5.

## 2.3 Cascade System Overview

Cascade is an overlay network-based application level multicast system. However, its design incorporates several techniques for alleviating the performance limitations arising out of the mismatch between the overlay network and the underlying IP network. In this section we briefly discuss the design architecture of the Cascade system.

The high-level design of the Cascade is similar to the SCRIBE [17]. Due to space limitations, our discussion about the architectural design of Cascade is compact. However, we highlight the key differences between the SCRIBE system and our design at appropriate locations.

### 2.3.1 System Architecture

The various nodes in the Cascade system interact with one another in a peer-to-peer fashion. An individual node can act both as a client and as a server for multicast requests. Henceforth we refer to the nodes of the Cascade system as *peers*. Every peer participates in multicast execution. Any peer can create a new multicast service of its own interest or subscribe to an existing multicast service. Our design of the Cascade system does not require any peer to have global knowledge of about other peers, or about all the multicast services that are currently being offered. Further, the peers can enter and exit the system at arbitrary points in time.

The peers in the Cascade system are organized as a *structured* P2P network, based on the concept of *distributed hash table (DHT)*. Each peer in our system is equipped with a Cascade middleware, whose design is depicted in Figure 2. The Cascade middleware is composed of two functional substrates: *P2P Network Management* and *ESM Multicast Management*.

The P2P network management substrate is the lower tier of the Cascade middleware, and it provides services for P2P network membership management, resource lookup, and communication among end-hosts. For example, a peer invokes the services provided by this layer to enter the network, or to communicate with another node in the network.

The ESM management substrate is responsible for ESM event handling, multicast group

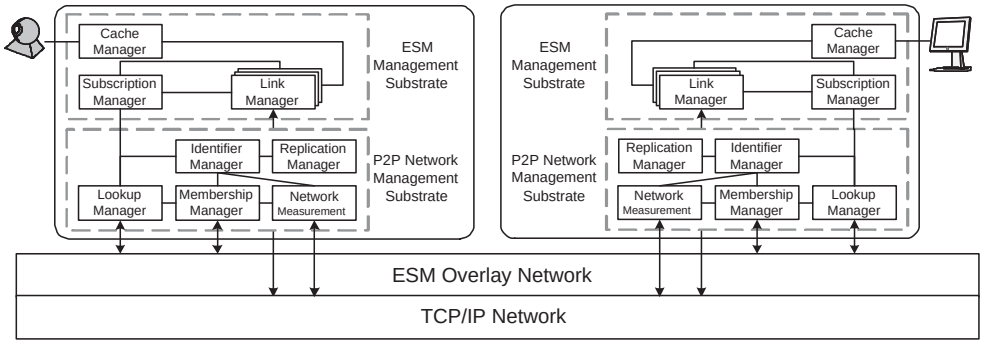


**Figure 2:** Cascade system architecture

membership management, multicast information delivery, and cache management. This layer utilizes the services provided by the underlying network management layer and incorporates three protocols namely, *multicast group membership management protocol*, *multicast information dissemination protocol* and *multicast overlay maintenance protocol*. The multicast group membership management protocol provides mechanisms for the peers to create new multicast service, subscribe to an existing multicast group, or exit from an existing multicast group. As we discuss later in this chapter, this protocol maps each multicast service to an individual peer in the network, which acts as the proxy-source of the service. Further, the protocol incrementally builds a multicast tree from subscribers of a multicast service. The multicast group membership management protocol propagates the multicast payload through this tree, so that it reaches all the subscribers. The multicast overlay maintenance protocol handles the exit and failure of peers in a multicast tree by appropriately repairing the tree.

In the Cascade system each multicast service has two unique $m$-bit identifiers associated with it, namely the *service identifier* and the *group identifier*. The *service identifier*, represented as $S_{id}$, uniquely identifies the multicast service and would be used to publish summary information about the multicast service. The group identifier $g_{id}$ will be used to identify the group of peers subscribed to the service.

Each peer in the Cascade system is also assigned a set of $m$-bit identifiers, which we henceforth call *peer-identifiers (Peer-IDs, for short)*. As we will explain later, when a peer $p$ wants to start a new multicast service, it will use one of its peer-IDs as the group identifier

of the new service. For ease of understanding, let us for now assume that each peer is associated with a single identifier.

The Cascade system is based on a structured P2P architecture. Hence it does not require any node to maintain global information about all the peers in the system at any point of time, or about all the services being offered. Instead, each peer maintains information about a carefully chosen subset of peers and multicast services currently available in the Cascade system. This distributed information would be used by the peers to lookup other peers, discover multicast services, subscribe to multicast services, and to build and maintain multicast trees.

In order to better understand the protocols and mechanisms of the Cascade system, we can conceptualize the peer identifiers, multicast service identifiers, and the multicast group identifiers as points on a logical ring with range from 0 to $(2^m - 1)$. With this conceptual model as the basis, we will first define a few terms, which will be used to explain various aspects of the design architecture of the Cascade system. The distance between two identifiers $i$, $j$, denoted as $Dist(i, j)$, is the shortest distance between them on the identifier circle, defined as $Dist(i, j) = min(|i-j|, 2^m-|i-j|)$. Identifier $i$ is considered as *numerically closest* to identifier $j$ when there exists no other identifier with a shorter distance to $j$, i.e. $\forall_{k \neq j} Dist(k, j) \geq Dist(i, j)$. A peer $p'$ with its peer identifier $j$ is said to be an *immediate right neighbor* to a peer $p$ with its peer identifier $i$, denote by $(p', j) = IRN(p, i)$, if there are no other peers have identifiers in the clockwise identifier segment from $i$ to $j$ on the identifier circle. Analogously the peer $p$ is called the *immediate left neighbor* of peer $p'$.

Each peer in the Cascade system is equipped with two data structures for maintaining information about a specific set of other peers. Specifically, for each peer-ID that is owned by a peer $p$, it maintains two types of information, namely a *routing table* and a *neighbor list*. The routing information in these two data structures would be used to lookup a peer or a service. The routing table is used to locate a peer that is more likely to answer the lookup query, whereas a neighbor list is used to locate the peer that is numerically closest to the identifier of the lookup query.

**Routing Table:** We denote the routing table associated with identifier $i$ of peer $p$ as

$RoutingTable(p, i)$. The routing table stores a set of peer identifiers (that satisfy a specific condition discussed below) and their reference information (IP address and port) of the peers that own those identifiers. The peer $p$ can directly communicate to the peers that own any of the identifiers in the routing table. The $m$-bit peer identifiers are represented in the routing table in radix $2^b$, implying that each identifier will have $\lfloor m/b \rfloor$ digits. The routing table has $\lfloor m/b \rfloor$ rows and $\lfloor m/b \rfloor$ columns. The entries of the routing table satisfy two rules:

1. Entries in the $i$th row (rows start from 0) of a table contain pairs with peer identifiers sharing $i$ leftmost digits with the associated identifier of the peer owning the routing table.

2. An identifier in the $j$th column of the $i$th row of a routing table has $j$, as the $i$th digit from the left (leftmost digit is indexed as 0).

As an example consider an 8-bit identifier space and let $b = 2$. Now each entry in the routing table will have 4 digits. Figure 3 shows a possible routing table for a peer identifier 2103. Note that all the entries in the routing table obey the above stated rules.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0210 | 1302 | | 3002 |
| 1 | 2033 | | 2213 | 2301 |
| 2 | | 2113 | 2122 | 2132 |
| 3 | 2100 | 2101 | 2102 | |

**Figure 3:** An example routing table for the peer identifier 2103

**Neighbor List.** In addition to routing table, the peers also maintain a list of neighboring peer identifiers and their reference information. Specifically, for an identifier $i$, the peer $p$ owning $i$ keeps track of $r$ successor identifiers on the logical ring and $r$ predecessor identifiers on the logical ring. Again the peer $p$ can communicate directly to any of the peers owning the identifiers that are present in the neighbor list. For example, a possible neighbor list of peer identifier 2103 would be $[2101, 2102, 2103, 2113, 2120]$, when $r$ is set to 2 (again only the peer identifiers are shown here for simplicity).

The purpose of maintaining the above two routing information in the peers is to enable

them to lookup the service identifiers or the group identifiers of any multicast service in the system. Specifically, any peer in the system should be able to discover the peer identifier that owns a given service identifier or a group identifier, and the reference information (the IP address and the port) of that peer to which the peer identifier is assigned. Lookup is the most fundamental operation in the Cascade system, through which peers can advertise new multicast services or discover and subscribe to existing services. We now outline the algorithm for the lookup operation when a peer $p$ is attempting to lookup identifier $i$. The lookup process is composed of three steps. In the first step the peer initiating the request (peer $p$) selects an identifier from its set of peer identifiers (recall that peer may have multiple identifiers) that is closest to $i$ (in terms of the $Dist$ function defined above). Let us denote the closest peer identifier to be $j$. Now the peer $p$ determines whether the identifier $j$ owns the service identifier $j$. This can be done by looking at neighbor list and routing tables of $j$ to see whether there is a peer closer to $i$. If $j$ owns the identifier $i$, lookup terminates since we have found the owner. If $j$ does not own $i$, it means that there is at least one node either in the neighbor list or routing table of $j$ that is closer to $i$, in which case the algorithm proceeds to the second step. In the second step the algorithm determines whether any of the peer identifiers in the neighbor list of $j$ own the identifier $i$ (again this can be done by checking the consecutive nodes in the neighbor list). If we find a peer identifier that owns $i$, the algorithm terminates. Otherwise, the algorithm executes the third step, wherein we select a peer identifier from $j$'s routing table that is closer to $i$ than the peer identifier $j$. Let this identifier be denoted as $l$. The lookup query containing $i$ is now forwarded to the peer which has $l$ as one of its identifier. This peer on receiving the forwarded lookup request performs the second and third steps, and the query might be forwarded to a third peer. Eventually the algorithm terminates and the peer identifier owning $i$ would be found. It can be shown that in this algorithm the number of times a lookup is forwarded is at most $O(\log N)$, where $N$ is the total number of nodes in the system. We will illustrate the lookup process with an example when we discuss the multicast mechanism in our system.

The Cascade system also includes algorithms to initialize the routing table and the neighbor list when a peer joins the Cascade system, and to modify them appropriately

when new nodes enter the system or existing peers exit the system. For the purpose of brevity and succinct of presentation, we omit the discussions on these schemes. Details about them can be found in the technical report version of this chapter [63].

### 2.3.2 Basic Multicasting in Cascade system

In Cascade, multicast service establishment and maintenance occurs through three distinct operations.

**Step 1: Publishing the Multicast Service:** Multicast sources join the Cascade P2P overlay as peers. As mentioned earlier, each multicast service is associated two identifiers, namely, service identifier and group identifier. The service identifier will be used to advertise and publish meta-information about the service, whereas the group identifier would be used by peers to subscribe to and unsubscribe from the multicast service. A peer generates these two identifiers for each multicast service for which it is a source. Suppose a peer $p$ wants to initiate a new multicast service $S$, it selects one of its unused peer identifiers and uses it as the group identifier of the multicast service $S$. Contrastingly, the service identifier $S_{id}$ will be generated by replacing a potion of one of the peer identifiers of $p$ with a number obtained through a certification service.

After generating the two identifiers for the multicast service, the source will publish the summary of the multicast service and its group identifier on another peer in the system. The node which hosts the summary of the multicast service $S$ is called $S$'s *rendezvous node*. The rendezvous node is determined through the lookup protocol discussed above. The source peer initiates a lookup query on $S_{id}$, which discovers the rendezvous peer in at most $O(\log N)$ hops. The source node publishes the summary and the group identifier of the new multicast service on the discovered rendezvous node. Any other node in the system can now obtain summary and the group identifier of the multicast service by doing a similar lookup on the service identifier of the multicast service, and contacting the respective rendezvous node.

**Step 2: Multicast Tree-based Subscription Management:** Now let us see how a peer can subscribe to a multicast service that it is interested in. As described above a peer can lookup the rendezvous node of any existing multicast service and obtain its group identifier. Since the group identifier of a multicast service is always chosen from one of the source peer's identifiers, the lookup operation (described in Section 2.3) always maps the group identifier to the source, irrespective of the scale or the dynamics of the network. Hence, an arbitrary node in the system can locate the source node of a multicast service given the service's group identifier.

The newly subscribing node (denoted by $p$) issues a subscription request with the group ID of the multicast service. This subscription request is treated exactly like a lookup request on the group ID, and is forwarded towards the multicast source through a series of intermediate peers, whose identifiers satisfy the progressive prefix matching criterion. At each forward step, the peer receives the subscription request records the sender as its child node in the multicast tree, before it forwards the subscription request another peer whose peer-ID is closer to the group ID. However, the request may not always reach multicast source. If one of the intermediate nodes has already subscribed to the multicast service requested by $p$ (i.e. the node is already in the multicast tree), the forwarding of the multicast subscription request is terminated. If none of the intermediate nodes are in the multicast tree, the request reaches the multicast source. All the peers handled the subscription request compose a multicast path connecting $p$ into the multicast tree. An example is given by Figure 4. Peer 0201 tries to subscribe to a multicast service identified by a group ID 2123. The subscription request is forwarded in sequence by peer 2332, peer 2102, and peer 2121, and finally reaches peer 2122 who is the root of the multicast tree identified by group ID 2123. The path $2122 \longrightarrow 2121 \longrightarrow 2102 \longrightarrow 2332 \longrightarrow 0201$ will be used to deliver the multicast contents to the subscriber peer 0201.

Conversely, the multicast tree is pruned when nodes *unsubscribe* from the respective multicast service. A node $p$ that wants to unsubscribe from a multicast service that it is currently member of, intimates its parent in the corresponding multicast tree through *unsubscribe request*. On receiving an unsubscribe request, the parent node, say $q$, removes

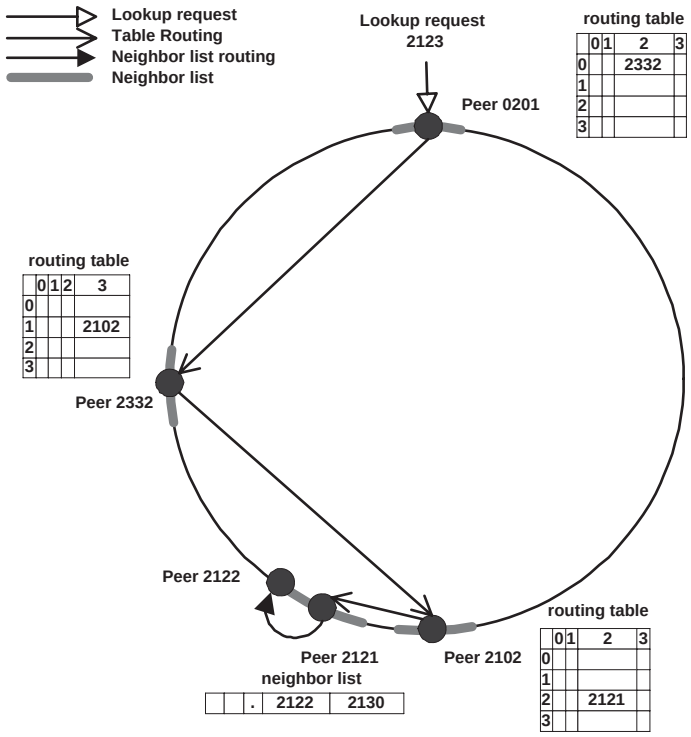


**Figure 4:** An example of Cascade subscription process

the node from the multicast tree. Further, $q$ also checks whether it has other children who are subscribed to the same multicast service. If there are no other nodes that are receiving the multicast service through $q$, and $q$ itself is not subscribed to the multicast service, it sends an unsubscribe message to its parent.

**Step 3: Dissemination of Multicast Payload** The source of a multicast service uses the corresponding multicast tree for delivering the multicast data to all the subscribers. It injects the data at the root of the multicast tree, which then gets disseminated through the tree, replicated at each intermediate node, and reaches all the subscribers.

### 2.3.3 Limitations of the Basic End System Multicast

The above end system multicast scheme has three limitations that can hinder its performance. These limitations have to be adequately addressed in order to ensure the scalability, efficiency, and reliability of the Cascade system.

The first limitation arises from the mismatch between the P2P overlay network topology and the underlying physical network topology, which has been well studied in prior literature [45, 49, 60]. In most generic P2P networks, the indexing techniques and the routing schemes are completely independent and oblivious to the underlying network structure. Hence, communication in these networks is likely to be very inefficient in terms of the physical network route traversed by individual message.

Since the multicast tree construction in the Cascade system utilizes the lookup mechanism of the P2P network (see Section 2.3), the mismatch between overlay network and the underlying physical network adversely affects the multicast performance. The overlay-underlay mismatch manifests in the Cascade system in the following manner. The multicast tree that is constructed at the logical level can be very inefficient in terms of the physical network connections. Consider the multicast tree depicted in Figure 5. In this example, six end-hosts located in three states participate an ESM overlay. Each of them is denoted by the state acronym plus an index number. Node WA 1 serves as the multicast root and all others are subscribers. Since the multicast tree is constructed without considering the physical network, the multicast messages have to travel from WA 1 (located in Washington) to GA 2 (located in Georgia) and then again traverse the link from GA 2 to WA 2. Thus, the multicast messages have to traverse between the east coast and west coast twice and three times along the east coast. This not only affects the multicasting latency, but also increases the load on the underlying physical network. Instead, if the nodes were to be organized in a tree as shown in Figure 6, the multicast would be significantly more efficient as the multicast messages do not have to traverse to and fro between the nodes located at the two coasts. However, because the ESM scheme is oblivious to the underlying physical network, it cannot ensure efficiency of the resultant multicast tree.

The second shortcoming of the basic ESM mechanism is the load imbalance between the participating nodes. One of the main causes of load imbalance is the heterogeneity in the resource availabilities at various nodes. Ideally the load on each peer should be proportional to its resource capabilities. This would avoid nodes becoming bottlenecks, thereby improving the efficiency and dependability of the system. Therefore, it is essential


**Figure 5:** Less efficient multicast overlay topology

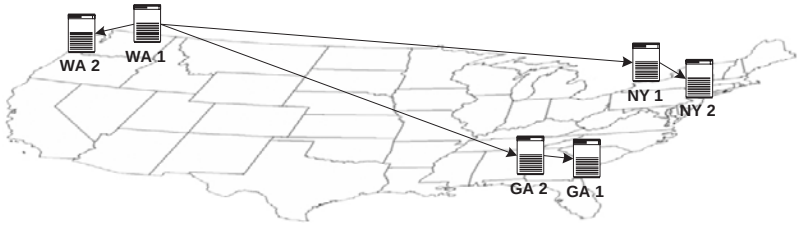


**Figure 6:** Grouping end-hosts by network proximity

to augment the basic ESM multicast scheme with techniques that distribute the multicast load among the various nodes in the system in accordance with their resource availabilities.

The third limitation of the basic ESM multicast scheme is that its timeout and resubscription-based mechanism to repair multicast trees that are damaged due to the failure/exit of peers are often ineffective. This drawback becomes especially pronounced when the P2P network is highly dynamic, which is the case in most P2P systems. Therefore, we need to design mechanisms that can guarantee reliability of the Cascade system, even when the P2P network exhibits considerable churn.

The remainder of this chapter is dedicated to design of effective and efficient mechanisms to overcome each of the above limitations of the basic ESM scheme.

## *2.4 Network Aware Multicast Tree Construction*

Our discussion in Section 2.3.3 highlighted the need and importance of taking the physical network locations of the nodes into consideration while building multicast tree. However, developing such multicast mechanisms poses two significant research challenges. The first is to devise techniques for accurately estimating the relative locations of the nodes in the physical network. Due to the decentralized and highly dynamic nature of Cascade network, it is not possible to maintain a complete global view of the overlay network or the underlying

network. The second major challenge is to incorporate the network position awareness into the decentralized framework of the ESM multicast tree construction framework.

Prior works such as Pastry [49] and CAN [45] have explored various techniques for improving the network awareness of P2P systems. In Pastry [49], peers probe the network neighbors and carefully select entries to fill into their routing tables. A list of network neighbors is setup along with the routing table to accelerate the routing. However, the Pastry approach has two major problems. First, its assumption about the triangular inequality properties in the IP network may not always hold. Second and more importantly, due to "logarithmic routing deterioration " [60], the benefits obtained by this scheme might be extremely limited. The CAN system [45] tries to map a peer into the CAN identifier space by its network coordinate information. This approach may introduce uneven identifier distribution in CAN hypercube space, thus resulting in some routing paths that are of poor quality.

We now explain our approaches to address the above two challenges. In order to address the challenge of estimating relative locations of the nodes, we use technique that is similar to the concept of *landmarks*. Conceptually, Internet landmarks (landmarks, for short) [21, 2, 54] are a set of few key Internet hosts that serve as a frame of reference for determining the relative position of any other node on the Internet. An arbitrary node measures the round trip time to each of these landmarks, and uses these values to determine its relative location in the physical network.

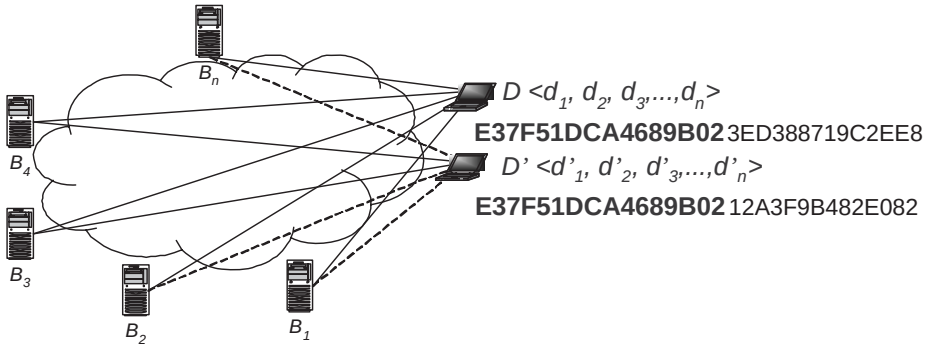


**Figure 7:** Using landmark to generate peer identifiers

Concretely, let us denote the landmark set as $\{B_1, B_2, B_3, \ldots, B_n\}$. When an end-host

joins the network, it obtains the landmarks set through the bootstrapping service. An end-host measures its distance to the given set of landmark points and records the results in a vector $D < d_1, d_2, d_3, \ldots, d_n >$, which we refer to as its *Landmark Vector*. The intuition behind using landmark vectors is that the end-hosts that are close to one another will have similar landmark vectors, as their probing packets are more likely to traverse similar network routes to reach each landmark point.

Designing a completely decentralized multicast tree construction technique that seamlessly utilize the above information about relative locations of nodes is a more difficult challenge. Notice that the basic ESM multicast tree construction scheme is indeed completely distributed and decentralized. We design two unique techniques to augment the basic ESM mechanism so that the resultant multicast tree is sensitive to the relative locations of the nodes thereby optimizing the communication costs.

First, we have designed a novel landmarks-signature technique to embed network positions information of each node into its identifier. Recall that the basic ESM scheme is based upon the lookup scheme, which in turn is based upon the identifier distribution of the nodes and the multicast service. By embedding network position information into the node identifiers, we try to ensure that the generated multicast tree is efficient in terms of the communications costs. The basic idea is to allocate identifiers such that the peers physically closer would have numerically closer peer identifiers. The landmarks signature of a node is based upon its landmarks vector. Specifically, we utilize the relative order of the elements of a node's landmark vector to capture the location of the node. We encode this relative order of landmark vector elements into a binary string and call it as the *landmark signature* of a peer. Therefore peers that are located in close network proximity tend to have numerically similar landmarks signatures. Thus, by comparing the landmarks signature of two nodes one can infer their proximity within the network. Figure 7 gives an example of how two physically close end system nodes generate their landmark signatures and P2P identifiers.

One question that arises is: *why not use the landmark signature of a peer as its identifier?* Previous studies have shown the importance of preserving enough randomness of the identifier distribution [60]. Hence, we see that the identifiers of nodes have to satisfy

two conflicting requirements. On one hand they should encapsulate the node's position information and on the other it should also have enough randomness. In the Cascade we balance these conflicting requirements as follows. Upon entering the network, a new peer generates its identifiers using the normal identifier generation functions such as MD5 and SHA-1. Further, it also measures its distance to various landmarks and generates its landmarks signature. Now, it replaces a substring of the generated identifier with the landmark signature at a certain offset. We call this offset as the *splice offset*. Its value is a system parameter that could be tuned according to the overlay population. The modified identifiers now contain information that can be used to identify the network location of the new peer, while preserving randomness to some degree.

The peer identifiers can be used to cluster peers based on their network proximities. Suppose the splice offset is set to $s$ digits. We can now envision the leading $s$ digits before the splice offset to randomly partition the identifier space into $2^{bs}$ "buckets". By controlling the value of the splice offset, we uniformly scatter peers from the same network proximity into those identifier "buckets". Peers of similar network proximity are still clustered together within each "bucket" by their landmark signatures. But they will not occupy a large continuous P2P identifier segment. The advantage of this approach is that it reduces the probability of the P2P network getting partitioned by the network domain level failure, which would forces a large number of peers to depart at the same time. The peer clustering improves the multicast performance as follows. As mentioned earlier, in our scheme the peers that close to one another are assigned identifiers similar identifier prefixes. But due to splice offset, these peers may be in different buckets. When a subscription request is forwarded among a number of peers sharing the same prefix, there is much higher probability that this request is forwarded among physical network neighbors. Thus, when we build the multicast tree using the lookup method of Cascade P2P protocol, we effortlessly achieve higher efficiency at the top portion of the multicast tree. This is because nodes located at the top of the tree share longer identifier prefixes with the multicast root node.

While our landmark signature technique improves the efficiency of the top portion of the resultant multicast tree, enhancing the efficiency of the bottom portion (portion near

the leaves of the tree) requires additional mechanisms. Towards this end, we develop our second technique called *Neighbor Lookup*. As suggested by its name, each peer initiating or forwarding a subscription request will first query its P2P network neighbors to see if they are already in the requested multicast tree. The closest neighbor in the multicast tree is chosen as the "potential parent". To avoid the looping of subscription request among neighbors, a peer compares its distances to its potential parent and grandparent. The subscription request will be forwarded to the closer one of them. The neighbor lookup is essentially a local operation in Cascade, because P2P neighbors frequently exchange status information to maintain the integrity of the P2P routing tables. Thus, a peer can learn the subscription status of its P2P neighbors by only checking the local cache of their status.

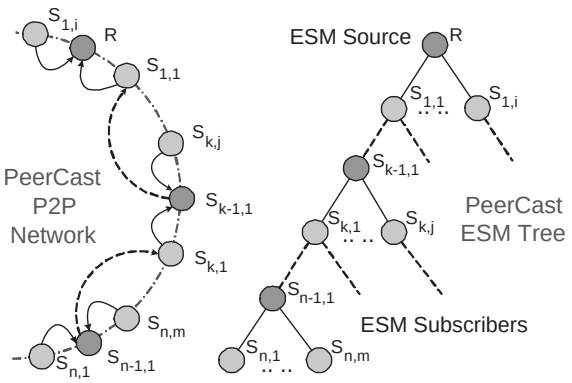


**Figure 8:** Improve Cascade ESM overlay with network proximity information

Figure 8 gives an example showing how the neighbor lookup technique works. Before forwarding the subscription request to the next hop peer that satisfies the prefix matching, peer $S_{k,1}$ first check if its neighbor has already joined the multicast group. It finds that peer $S_{k-1,1}$ is in the multicast tree and chooses it as its potential parent. Note that instead of forwarding its subscription request along the normal subscription path (denoted by the dotted lines) $S_{k,1}$ directly subscribe to peer $S_{k-1,1}$. Similarly, peer $S_{n,1}$ subscribes to its physical network neighbor $S_{n-1,1}$.

Our landmarks signature scheme ensures that a node present in a peer's neighbor list is also close to the peer in the physical network. Thus, in the multicast tree constructed using neighbor lookup technique, end-hosts close to one another are grouped together. One peer

in each group (the darker nodes in Figure 8) serves as the parent of other peers (the lighter nodes in Figure 8), and forwards the multicast payloads to them. Because the unicast links among network neighbors usually have higher bandwidth and lower latency, we improve the efficiency of the multicast tree and reduce the number of IP packets sent through the underlying network.

## *2.5 Load Balance in Cascade*

We now describe our techniques for addressing the second major limitation of the basic ESM scheme, namely, the load imbalance due to node heterogeneity. Measurement studies [50] have conclusively shown that end-hosts exhibit noticeable heterogeneity in large-scale P2P networks. For a distributed system in which end-hosts rely on one another to provide services like end-system multicast, balancing workloads among heterogeneous end-hosts is vital in utilizing the full system capacity and to provide efficient services. One way to address the node heterogeneity problem is to place end-hosts in a P2P network into different service layers depending upon their capabilities. A number of systems have adopted this approach to achieve better performance. For example, KaZaA [5] and Gnutella v.0.6 [58] have the notions of *super node* and *ultra peer* respectively. In the structured P2P network proposed by Zhao et al. [65], these nodes are referred to as *super nodes*. Xu et al. [59] organize the super nodes into a special layer of overlay called *expressway*, which is used to accelerate the routing and to provide better services. Generally, the powerful nodes are assigned with more workloads and serve as the "hubs" in the overlay network.

However, the above approach has one significant drawback. Because of the predetermined hierarchical system architecture, such schemes introduce vulnerabilities into the overlay network. Usually, a supernode does not use lower level nodes to relay traffic to other supernodes in the network. Rather, supernodes interact directly with one another. Meanwhile, supernodes are assumed to be stable and have enough resources such as computing power and access network bandwidth to serve their duty. The supernodes might unexpectedly drop out of the network in an uncontrolled manner due to node failure, forceful removal, or overloading. The overlay network would suffer considerably because it might be

fragmented into a number of disconnected smaller network partitions. The system would also be vulnerable when malicious nodes assume the role of supernodes and mislead other overlay nodes into relying on them for services.

The Cascade system adopts an alternative approach. Our approach is based on the concept of *virtual nodes*. A virtual node is a conceptual entity that encapsulates all the functionality of an individual peer. In fact, each virtual node has its own identifier in the Cascade system. An actual peer in the Cascade system is assigned one or more virtual nodes based upon the resource availability at the peer. In other words, an actual peer in our system *hosts* multiple identifiers and is responsible for performing the functionalities of the corresponding virtual nodes. For example, a node with higher capabilities is assigned larger number of virtual nodes, and vice versa. Thus, our scheme implicitly assigns more workloads to the powerful nodes.

The Cascade system incorporates three load-balancing operations:

**Generate Virtual Nodes.** Each end-host joins the Cascade ESM overlay with a set of identifiers generated with different random seeds. Each identifier represents a virtual node that is allocated with a unit of resource. We assume that the amount of resources that an end-host shares can be estimated by using functions like the one used in [31], or be specified by the end user. Each virtual node maintains its own overlay state information like routing table and neighbor list.

**Subscribe with Virtual Nodes.** An end-host subscribes to a multicast group by starting the subscription process at one of its virtual nodes with an identifier numerically closest to the service identifier $g_{id}$. Statistically, a more powerful end-host should have higher probability to own an identifier that is closer to any service identifier.

**Virtual Nodes Promotion** Our virtual node scheme assigns shorter multicast path to more powerful end-hosts with higher probability. However, because we use network proximity information to cluster end-hosts, and use randomly generated leading digits to control the distribution of identifiers, there is still a nonzero probability of a weak end-host

owning an identifier that is numerically close to the multicast root and thus be placed closer to the root of the multicast tree. Hence, it has to serve a large number of subscribers and it is likely to become a bottleneck.

One way to mitigate this problem would be to place only powerful nodes closer to the multicast root. Towards achieving this objective, we design a technique called *virtual node promotion*. In this technique, each node in the multicast tree periodically probes its child nodes. It chooses the child that has the most available resources as its potential replacement. Whenever a node detects that its potential replacement has more available resources than itself, an up-call is triggered. It will inform its potential replacement to subscribe to its parent, and inform its children to subscribe to the potential replacement. On receiving the promotion notification, the potential replacement will inform its children to subscribe to its current parent. The whole process is transparent to the end users.

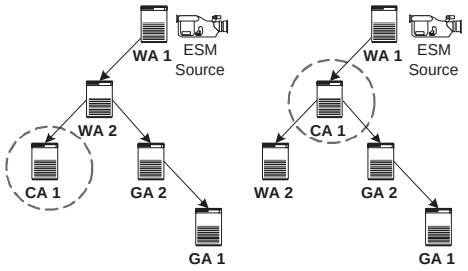


**Figure 9:** Bottleneck removal

Thus, end-hosts contributing more resources will be gradually "promoted" towards the root of the ESM tree and they obtain better multicast service than other end-hosts. Figure 9 gives an example, illustrating how this technique works. In the ESM overlay composing end-hosts {CA1, GA1, GA2, WA1, WA2}, WA2 probes its children CA1 and GA1. As WA2 detects that CA1 has more available resources than itself, it initiates the promotion of CA1 and switch its position to be a child of CA1.

### 2.5.1 Failure Resiliency in Cascade

Failure resiliency is the capability of a system to tolerant unintentional faults and non-malicious failures. In the Cascade system, a failure is represented by the interruption in

the multicast service to the nodes. It is typically caused by the exit of one or more peers in the system. Node exits can be of two types. A node might *properly depart* from the system, in which case, it voluntarily disconnects from the system. Here, the departing peer notifies the other peers about its departure. In contrast, a *failure* is a disconnection of a peer without notifying the system. This can happen due to a network problem, computer crash, or improper program termination. Failures are assumed to be detectable (a fail-stop assumption), and are captured by the Cascade P2P protocols neighbor list polling mechanisms. Irrespective of how a node exits, the mechanisms in Cascade to handle the node exit are similar.

One possible way to handle node exits would be to require the peers whose multicast service has been disrupted to re-subscribe to their respective multicast service [17]. This is in fact a fallback option in Cascade. However, this approach has a major drawback; each node exit would trigger large-scale re-subscriptions, which imposes significant overheads on the system.

The Cascade system incorporates a failure resilience mechanism that is based on the principle of service replication.

### 2.5.2 Service Replication Scheme

Our service replication scheme involves two phases. The first phase is right after the ESM group information is established on a peer. Replicas of the ESM group information are installed on a selection of peers. After replicas are in place, the second phase keeps those replicas in consistency as end-system nodes join or leave the ESM group.

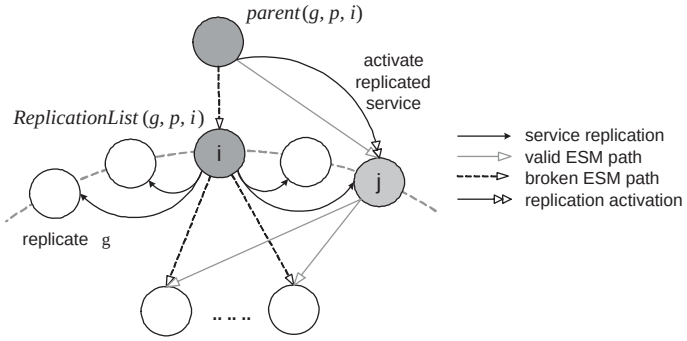


**Figure 10:** Multicast Service Replication with $r_f = 4$

In our scheme, the group information on a peer $p$ with identifier $i$ is replicated on a set of peers denoted as $ReplicationList(g, p, i)$. We refer this set as the replication list of group $g$ on peer $(p, i)$. The size of the replication list is $r_f$, which is referred as the replication factor and is a tunable system parameter. To localize the operations on the replication list, we construct the replication list from the neighboring peers of node $p$. This implies that $r_f \leq 2 * r$.

For each ESM group $g$ that a peer $p$ is actively participating, peer $p$ will forward the replication list $ReplicationList(g, p, i)$ to its parent peer $parent(g, p, i)$ in group $g$. Once $p$ departs from group $g$, its parent peer $parent(g, p, i)$ will use $ReplicationList(g, p, i)$ to identify another peer $q$ with identifier $j$ to take over the ESM multicast forwarding functionalities of $p$. $q$ will use the group information that $p$ installed on it to carry out the ESM forwarding for group $g$. We say that $q$ is *activated* in this scenario. Once $q$ is activated, it will use its neighbor list $NeighborList(q, j)$ to setup the new $ReplicationList(g, q, j)$, and use it to replace $ReplicationList(g, p, i)$ on $parent(g, p, i)$, which is equivalent to $parent(g, q, j)$ now.

### 2.5.3 Replica Management

We now explain the process of maintaining the replicas when end-systems join or leave the Cascade system.

For the purpose of brevity, we assume that the replication factor $r_f$ is equal to $2r$. In case that $r_f$ is less than $2r$, our arguments still hold with some minor modifications to the description.

When a multicast group $g$ is added to the multicast group list on a peer $p$ with identifier $i$, it is replicated to the peers in the $ReplicationList(g, p, i)$. Cascade P2P protocol detects the later peer entering and departure event fallen within $NeighborList(p, i)$. Once such an event happens, an upcall is triggered by the P2P management protocol, and the replica management protocol will query the peers in $NeighborList(p, i)$ and update the replication list $ReplicationList(g, p, i)$. We describe the reaction that a peer will take under different scenarios.

**Peer Departure.** The departure of a peer triggers the update of $2r$ neighbor lists. Once a peer $p$ with identifier $i$ receives the upcall informing the departure of peer $p'$, it will perform the following actions:

- For each group $g$ that $p$ is forwarding ESM payload, $p$ adds $p''$, which is added into $NeighborList(p, i)$ by the P2P management protocol, to the replication list $ReplicationList(g, p, i)$.
- For each group $g$ that $p$ is forwarding ESM payload, $p$ removes the departing peer $p'$ from the replication list $ReplicationList(g, p, i)$.
- For each group $g$ that $p$ is forwarding ESM payload, $p$ removes the departing peer $p'$ from the replication list $ReplicationList(g, p, i)$.
- For each group $g$ that $p$ is forwarding ESM payload, $p$ removes the departing peer $p'$ from the replication list $ReplicationList(g, p, i)$.

**Peer Entrance.** A peer's entrance also triggers the update of $2r$ neighbor lists. Once a peer $p$ with identifier $i$ receives the upcall informing the entrance of peer $p'$, it will perform the following actions:

- For each group $g$ that $p$ is forwarding ESM payload, $p$ adds $p'$, to the replication list $ReplicationList(g, p, i)$.
- For each group $g$ that $p$ is forwarding ESM payload, $p$ removes peer $p''$, which is removed from $NeighborList(p, i)$ due to the entrance of $p'$, from the replication list $ReplicationList(g, p, i)$.
- For each group $g$ that $p$ is forwarding ESM payload, $p$ sends its group information to $p'$ as replicas.
- For each group $g$ that $p$ is forwarding ESM payload, $p$ sends the updated replication list $ReplicationList(g, p, i)$ to its parent peer $parent(g, p, i)$ in multicast group $g$.

**Updating Replicas.** As end-system nodes subscribe or unsubscribed from ESM groups, their subscription or unsubscription requests will be propagated up in the ESM tree and change the group information on some peers. Once the group information of group $g$ is changed on peer $(p, i)$. $p$ sends its group information to all the other peers in $ReplicationList(g, p, i)$.

### 2.5.4 Replica Selection Policy

As mentioned earlier, when a node $p$ exits the Cascade network, its parents in the multicast tree *activates* one of the nodes in $p$'s replication list and asks it to handle the forwarding functionalities of $p$. Our choice of the peer to be activated depends upon two distinct factors, namely *peer load factor* and *replication distance factor*. Intuitively, the peer load factor measures the willingness of neighboring peer $p_r$ to accept one more multicast forwarding workload considering its current load. The replication distance factor, on the other hand, is a measure of the network proximity of the peer $p_r$ to the failed peer $p$. A utility function called *ReplicaSuitability* combines these two factors into a single value. The parent of the failed peer $p$ in the replication list of $p$ that has the highest *ReplicaSuitability* value and activates it.

We define each of these factors as follows:

Let $p_f$ denotes a failed peer and $p_r$ denotes a replica holder of $p_f$ for multicast group $g$.

**Peer load factor** is denoted as $PLF(p_r)$. It is a measure of a peer $p_r$s willingness to accept one more multicast forwarding workload considering its current load. It is defined as follows:

$$PLF(p_r) = \begin{cases} 1 & \text{if} \quad p_r.load \leq tresh * \text{MAXLOAD} \\ 1 - \frac{p_r.load}{\text{MAXLOAD}} & \text{if} \quad p_r.load > tresh * \text{MAXLOAD} \end{cases} \tag{1}$$

**Replication distance factor** is denoted as $RDF(p_f, p_r)$. It is a measure of the network proximity of the peer $p_r$ to the failed peer $p_f$. $RDF$ is defined as follows:

$$RDF(p_f, p_r) = \frac{1}{ping_time(p_f.IP, p_r.IP)} \quad (2)$$

Let $UtilityF(p_f, p_r, g)$ denote the utility function, which returns a utility value for activate the service replica of peer $p_f$ and group $g$ on peer $p_r$. It is calculated based on the two measures given above:

$$UtilityF(p_f, p_r, g) = PLF(p_r) + \alpha * RDF(p_f, p_r)) \quad (3)$$

Note that we give more importance to the peer load factor $PLF$. For instance, the service replica on a peer that is very close to the failed peer will not be activated if the replica holder is heavily loaded. $\alpha$ is used as a constant to adjust the importance of replication distance factor with respect to peer load factor. For a lightly-loaded ESM overlay, we want to have a larger value of $\alpha$ since the probability that peers get overloaded is lower, and a more efficient ESM overlay is more desirable. In a heavily-loaded ESM environment, we may want to have lower value of $\alpha$, to guarantee the feasibility of the multicast plan first.

## *2.6 Experimental Results*

We have designed a simulator that implements the mechanisms presented in this chapter. We evaluate the Cascade system in three subjects using our simulator. We first study the effects of landmark signature technique on clustering end-hosts. Then, we evaluate how the efficiency of multicast overlay could be improved using the network proximity information. Finally, we study the multicast workload distribution and system performance under the load-balancing scheme.

We used the Transit-Stub model from the GT-ITM package [61] to generate a set of network topologies. Each topology consists of 5150 routers. The link latencies are assigned values using a uniform distribution on different ranges according to the type of the links: Unif(15ms, 25ms) for intra-transit domain links, Unif(3ms, 7ms) for transit-stub domain links, and Unif(1ms, 3ms) for intra-stub domain links. End-hosts are randomly attached to the stub domain routers and organized into P2P networks following the Cascade P2P network management protocol. Multicast group of various sizes are built over the P2P network

following the multicast group management protocol, featured with different optimization techniques.

We used the routing weights generated by the GT-ITM package to simulate the IP unicast routing. IP multicast systems are simulated by merging the unicast routes into shortest path trees.

### 2.6.1 Clustering by Network Proximity

This set of experiments examine the precision that landmark signature technique can achieve in clustering end-hosts by their network proximity. The metrics we use is the percentage of peers that have physical network neighbors in their local P2P neighbor lists. We simulate the P2P networks with $1 \times 10^4$ to $9 \times 10^4$ peers, and set the neighbor list size parameter $r$ to be 8, 12, and 16. We choose 0 and 1 as the splice offset values.

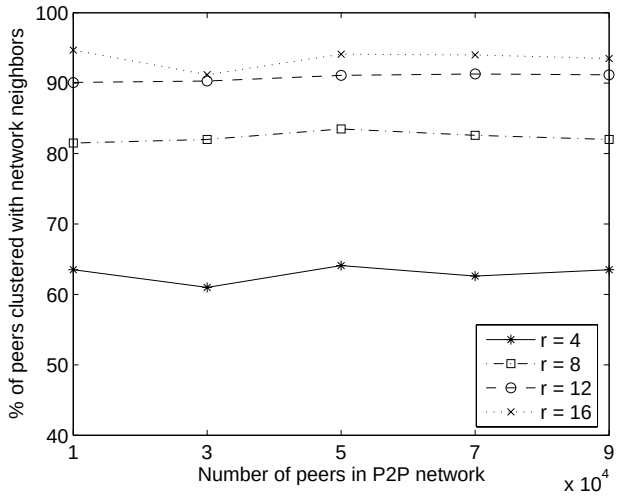


**Figure 11:** Clustering precision with *splice offset* = 0

Figure 11 and Figure 12 plots the experiment results. We observe three facts. First, larger value of $r$ increases the probability that peers find their physical network neighbors in their local P2P neighbor list. Second, the landmark signature scheme can capture the network proximity information with satisfactory precision. As many as 94% of all the peers have one or more network neighbors in their local neighbor list, when $r$ is set to 16. Third, larger peer population increases the precision of clustering, because more peers from the same network proximity join the Cascade ESM overlay.

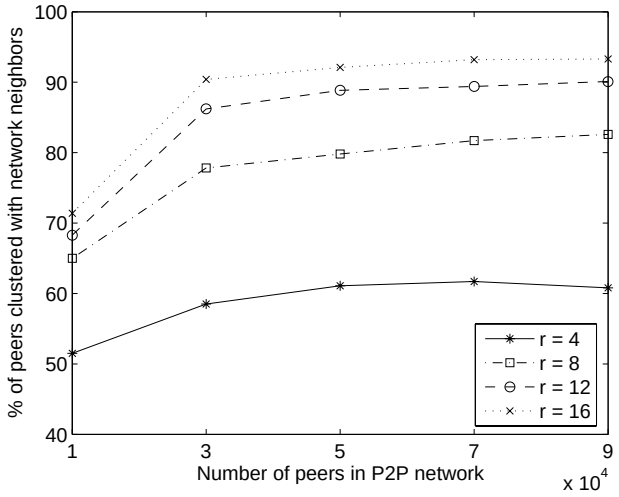


**Figure 12:** Clustering precision with *splice offset* = 1

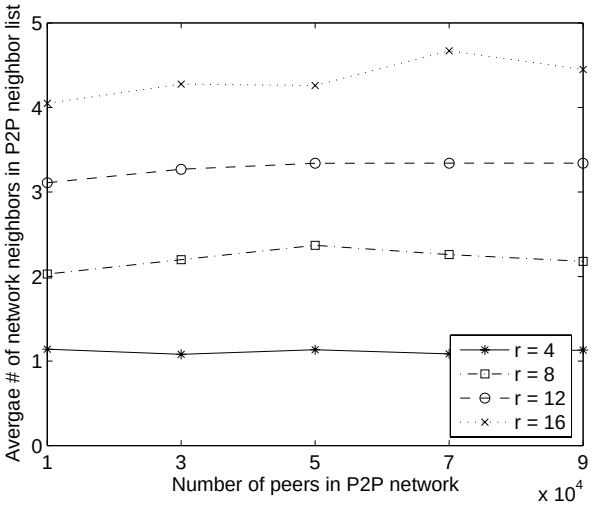


**Figure 13:** Clustering accuracy with *splice offset* = 0

The other metrics we used is the average number of physical network neighbors in each peer's P2P neighbor list. From Figure 13 and Figure 14, we first observe that the more peers participate the P2P network, on average more entries in a peer's P2P neighbor list are its physical network neighbors. Secondly, the size of the neighbor list is another factor that affects this number. As a peer's neighbor list covers more peers, it observes more physical network neighbors.

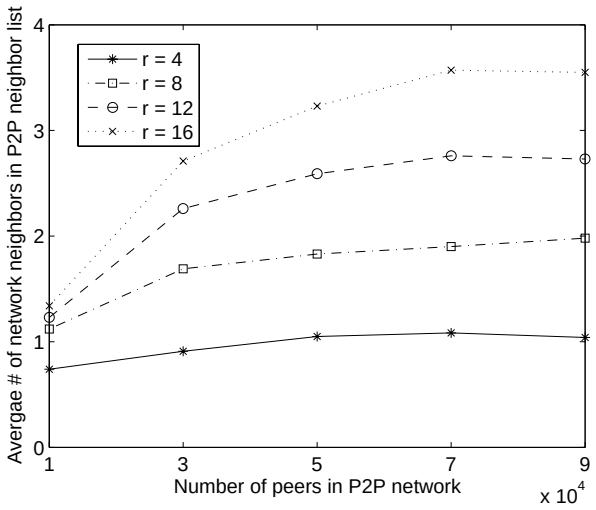


**Figure 14:** Clustering accuracy with *splice offset* = 1

### 2.6.2 Efficiency Multicasting in Cascade ESM Overlay

Because the landmark signature technique biases the distribution of peer identifiers, one of the concerns about using it is if it will incur any side-effects. In this section, we study three different flavors of the Cascade ESM system: the one without the landmark signature technique, which is equivalent to SCRIBE, the one with the landmark signature technique only, and the one with both the landmark signature technique and the neighbor lookup technique. We simulate a P2P network with $5 \times 10^4$ peers. The number of peers joining the multicast group varies from $1 \times 10^4$ to $4 \times 10^4$. We set the value of neighbor list parameter $r$ to 8 and used 16 landmark points to minimize the clustering inaccuracy, so that we can focus our efforts on comparing different schemes. We choose 0, 1, and 2 as the splice offset values.

#### *2.6.2.1 Delay Penalty*

We first compare the message delivery latency of IP multicast and Cascade. ESM systems have higher multicast message delivery latency than the equivalent IP multicast systems, because of the multi-hop message replication and unicast forwarding. We evaluate such performance penalty with metrics *Relative delay penalty*, which is defined as the ratio between the average Cascade multicast delay and the average IP multicast delay.

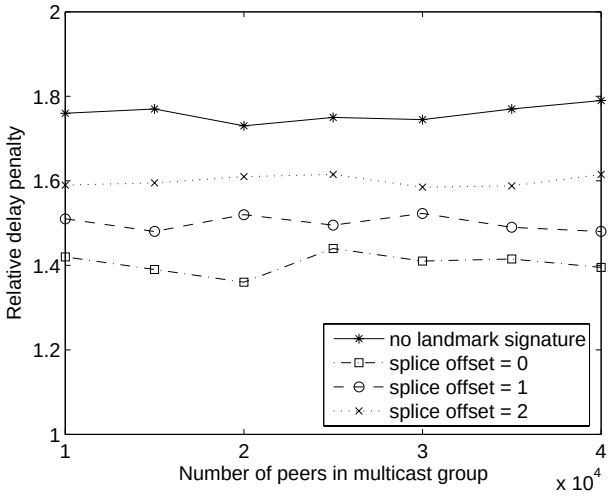


**Figure 15:** Relative delay penalty, using only landmark signature

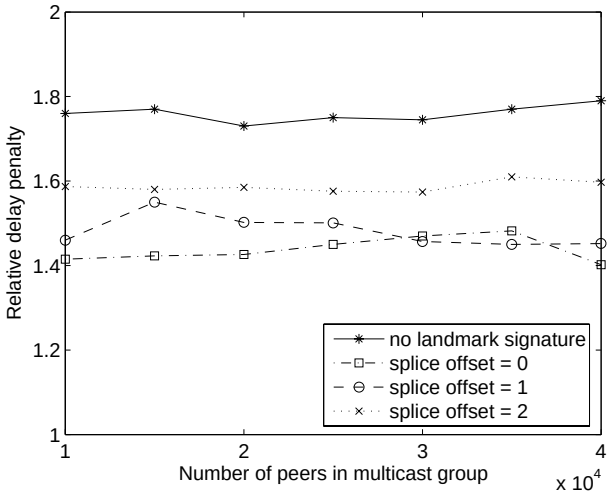


**Figure 16:** Relative delay penalty, using both landmark signature and neighbor lookup

The experiment results of Figure 15 and 16 show that, using the landmark signature technique alone can reduce the relative delay penalty. The landmark signature technique and our ESM management protocol put the network neighbors of multicast root node close to it in the ESM tree. Thus, the multicast delay of those end-hosts that receives multicast information through the network neighbors of root will have less delay penalty.

However, increasing the value of splice offset will offset the benefit of landmark signature technique. The randomness identifier distribution in Cascade P2P network degrades to the one similar to the random P2P network, when we increase the value of splice offset. The

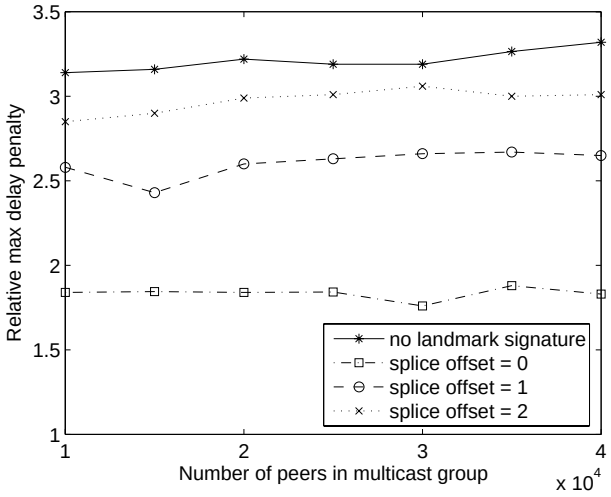


**Figure 17:** Max delay penalty, using only landmark signature

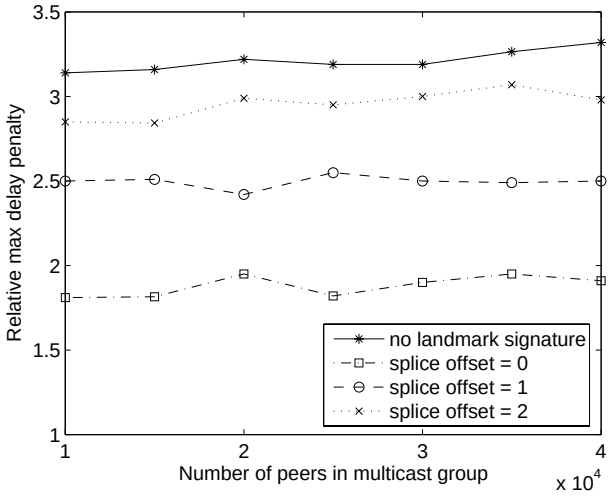


**Figure 18:** Max delay penalty, using landmark signature and neighbor lookup

Cascade systems with larger splice offset value have relative delay penalty values close to the ones of the basic Cascade systems.

Another observation is that the neighbor lookup technique has little influence on the relative delay penalty. It is because the landmark signature technique clusters peers by their network proximity. The nodes grouped by the neighbor lookup technique in the multicast tree are physical network neighbors of one another. The multicast communications among them are thus faster and add little to the overall multicast delay.

#### *2.6.2.2 Link Stress*

To reach the same set of end-host subscribers, ESM systems always generate more IP messages than the IP multicast systems. *Link Stress* is the ratio between the number of IP messages generated by a ESM multicast tree and the number of IP messages generate by the equivalent IP multicast tree. More efficient ESM systems have smaller link stress value.

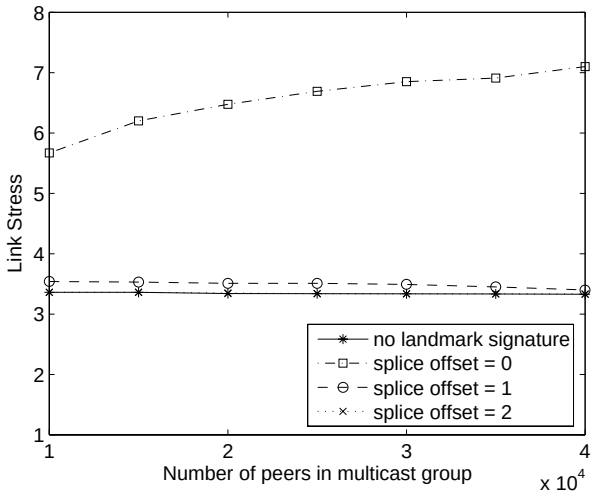


**Figure 19:** Link stress, using only landmark signature

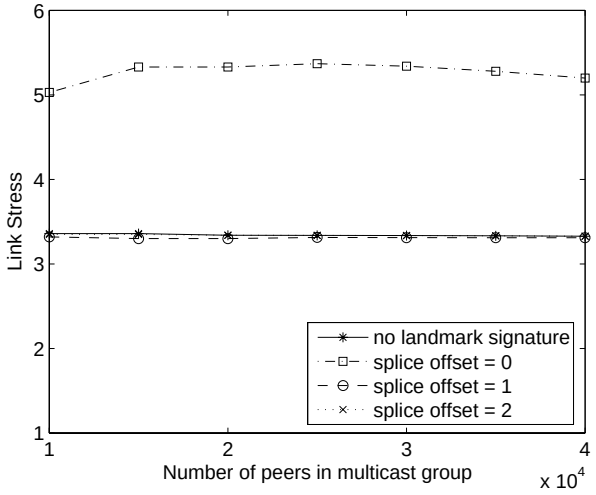


**Figure 20:** Link stress, using both landmark signature and neighbor lookup

The randomness of identifier distribution is violated by inserting landmark signature at the very beginning of each peer identifier. The first hop of lookup requests is longer because

the request initiator and the multicast root may not share the same identifier prefix, when they are not physical network neighbors. On the other hand, the routing paths are less likely to converge because they are more likely originated from different sub-networks, which are now represented by identifier clusters on different segments of the identifier circle. Hence, we observe that in Figure 19 and Figure 20, the serials with splice offset value 0 have much higher link stress than other cases. When we introduce more randomness into the identifier distribution by controlling the value of splice offset, we distribute peers from the same network proximity into different segments on the P2P identifier circle. Thus, the link stress drops close to the level of basic Cascade scheme, which is not optimized with the landmark signature technique and the neighbor lookup technique.

When we use the neighbor lookup technique in our multicast overlay, more multicast forwarding hops are confined among physical network neighbors. We thus reduce the number of multicast forwarding paths traversing the inter-network links. This mechanism reduces the link stress in comparison to the Cascade system without neighbor lookup scheme.

#### *2.6.2.3 Node Stress*

End-hosts in Cascade replicate and forward multicast messages, and handle housekeeping jobs like maintaining the multicast group membership information. We use *node stress* to evaluate the multicast workload on end-hosts. It is defined as the average number of children that a non-leaf end-host handles in the Cascade multicast tree.

The way we generate landmark signature reduce the dimension of identifier space by a factor of $L!/2^{bL}$, where $L$ is the number of landmark points. When we splice the landmark signature at the very beginning of each identifier, we reduce the number of peers that share the same identifier prefix with the multicast source. Subscription requests are then more likely to be forwarded through fewer peers in the P2P network, and consequently incur higher node stress. This explains why the Cascade ESM overlays with splice point 0 has much higher node stress in Figure 21. When we increase the value of splice offset, we reduce the node stress by increasing the randomness of identifier distribution. Thus, we see in Figure 21 that peers with splice offset of value 1 or 2 have link stresses close to the

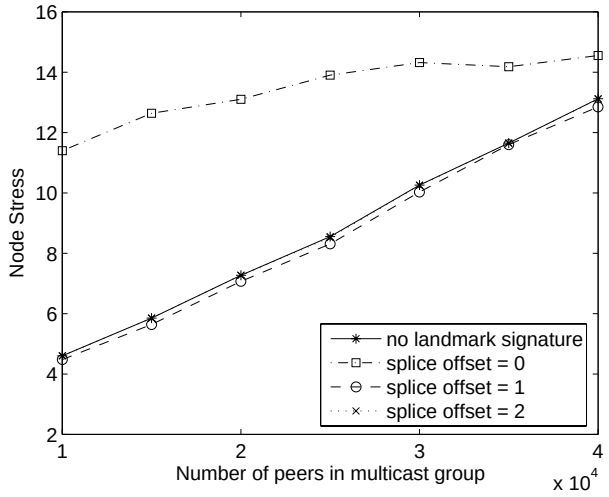


**Figure 21:** Node stress, using only landmark signature

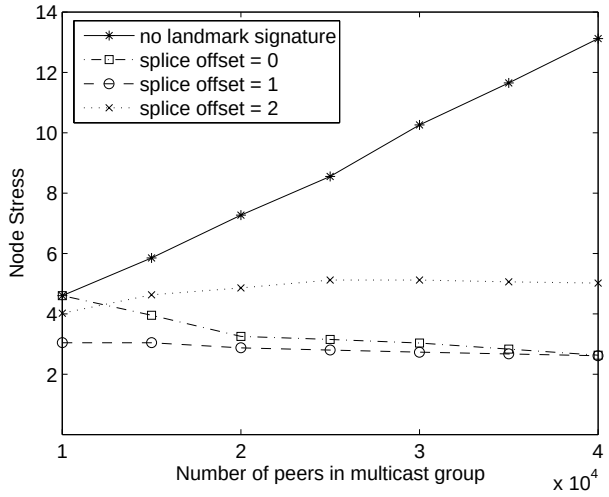


**Figure 22:** Node stress, using landmark signature and neighbor lookup

ones of basic Cascade scheme. As the number of end-hosts in the multicast group increases, in average a peer will handle more forwarding links, thus the average node stress increases accordingly.

Using the neighbor lookup technique, a peer first trying to leverage its local network neighbors for multicast services. The landmark signature technique renders high probability that a peer can identify a network neighbor in its local P2P neighbor list. Therefore, we observe in Figure 22 that Cascade ESM overlays equipped with both features have much lower node stress in comparison to the original Cascade scheme. As the number of peers in

the multicast group increases, the chance that a peer could subscribe to its local network neighbors increases too. This explains why node stresses decrease when the number of peers in the multicast group increases.

### 2.6.3 Balancing Load of Heterogeneous End-Hosts

To evaluate the effects of virtual node technique, we assign end-hosts in our simulation with different capacities. To have a clear measurement of multicast load, we assume that 20% end-hosts possess 8 units of capacity, 30% end-hosts have 4 units of capacity, and the rest 50% end-hosts own only single unit capacity. We are interested in the load distribution among end-hosts with different capacities. Also, we are interested in the quality of multicast services received by end-hosts donating different amount of resources. In Cascade, the quality of multicast services is measured by the relative delay penalty of different end-host groups.

We setup two types of ESM overlays. One set of them are built over P2P networks with fixed peer number $5 \times 10^4$ and various multicast group sizes, which model shared P2P networks. Another set of ESM overlays are built over P2P networks of various size. This model simulates exclusive P2P networks where only the provider or subscribers of the same multicast service join the P2P network. We varies the number of end-hosts in the ESM multicast overlay from $5 \times 10^1$ to $5 \times 10^4$. The node stress and the relative delay penalty are recorded for each group of end-hosts with different capacities.

Figure 23 and Figure 24 plots the average node stress of ESM overlays optimized with virtual node promotion technique. As shown in both figures, virtual node technique can match the multicast workloads to end-hosts according to their capacities. In Figure 23, where the multicast group members are chosen among shared P2P overlays, the node stresses of different end-host groups present less significantly differences because fewer end-hosts are heavily loaded. However, as recorded in Figure 24, when P2P overlay is heavily loaded, the virtual node technique plays a vital role in balancing multicast workloads. Because the basic Cascade protocol does not distinguish capacities of end-hosts, the randomness of the peer identifier distribution causes a number of powerful end-hosts being placed down into

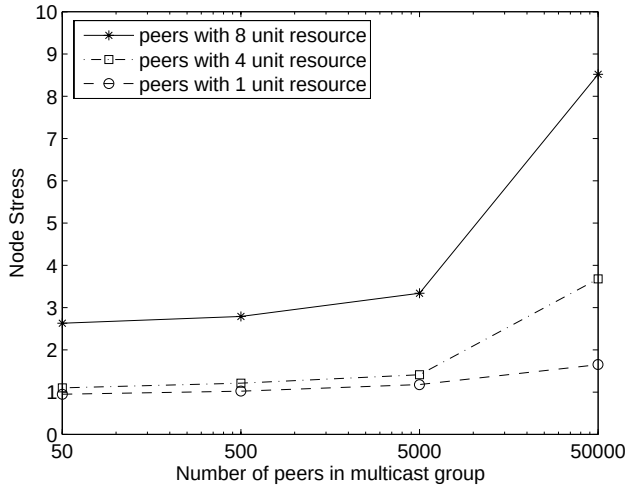


**Figure 23:** Average node stress, $r = 8$, peer number = 50,000

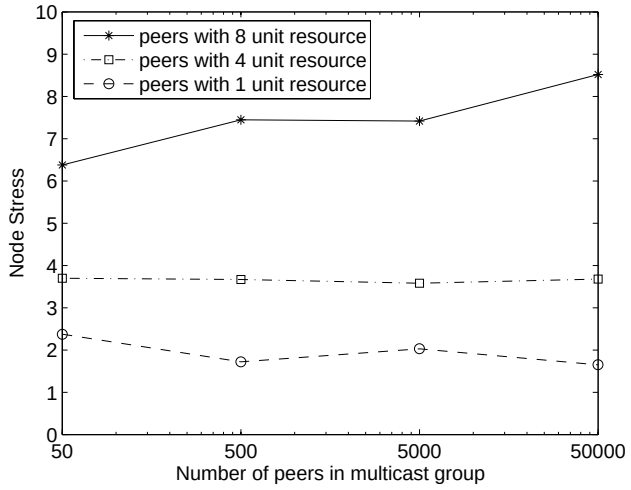


**Figure 24:** Average node stress, $r = 8$, peer number = multicast group size

the the ESM tree, whereas some less capable nodes are placed close to the root. The node promotion technique we discussed in Section 2.5 solves this problem by promoting more capable nodes towards the root node to handle more ESM workload, and moving the potential bottleneck nodes deep down into the tree. As plotted in both Figure 23 and 24, this feature effectively solves this problem.

One of the unique features of Cascade is that the end-hosts that contribute more resources will be placed closer to the multicast root peer. This feature not only assigns more

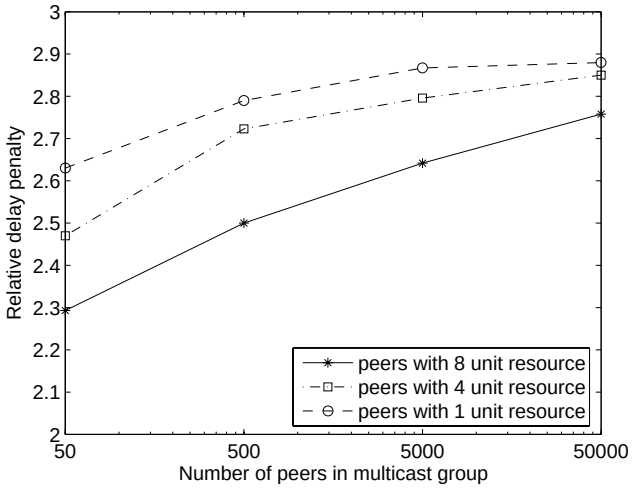


**Figure 25:** Relative delay penalty, $r = 8$, peer number = 50,000

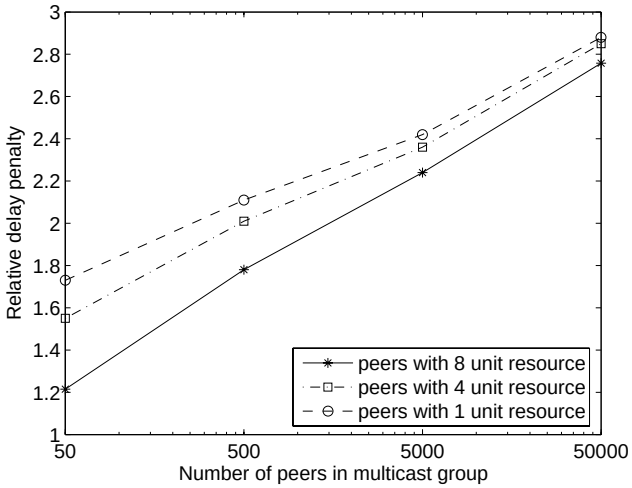


**Figure 26:** Relative delay penalty, $r = 8$, peer number = multicast group size

multicast workloads to more capable peers, it also awards them with better multicast services in term of lower relative delay penalty. We records the relative delay penalties of end-hosts with different level of capacities in Figure 25 and Figure 26. We can see that the more capable peers always have lower delay penalty.

### 2.6.4 Reliability

#### *2.6.4.1 Failure Resilience*

One of the situations that is rather crucial for the Cascade system is the case where the peers are continuously leaving the system without any peers entering; or the peer entrance rate is much lower than the peer departure rate such that the number of peers in the system decreases rapidly.

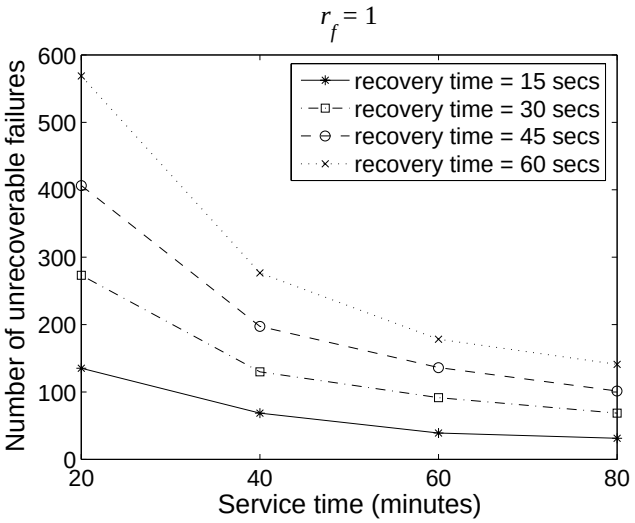


**Figure 27:** Unrecoverable failure, $r_f = 1$

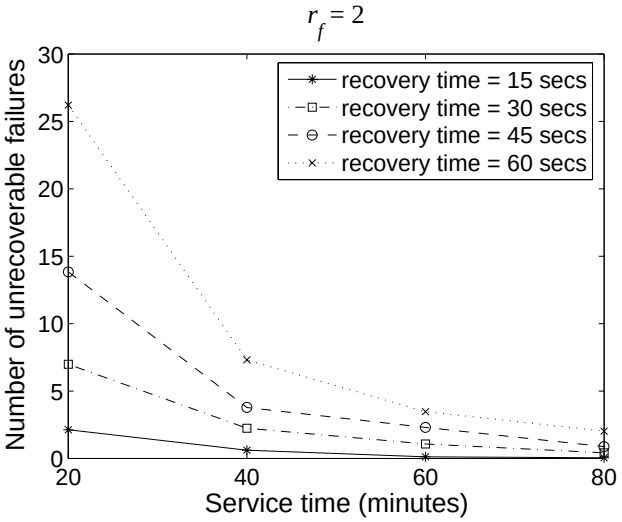


**Figure 28:** Unrecoverable failure, $r_f = 2$

To observe the worst case, we setup our simulation with $4 * 10^4$ peers, among which

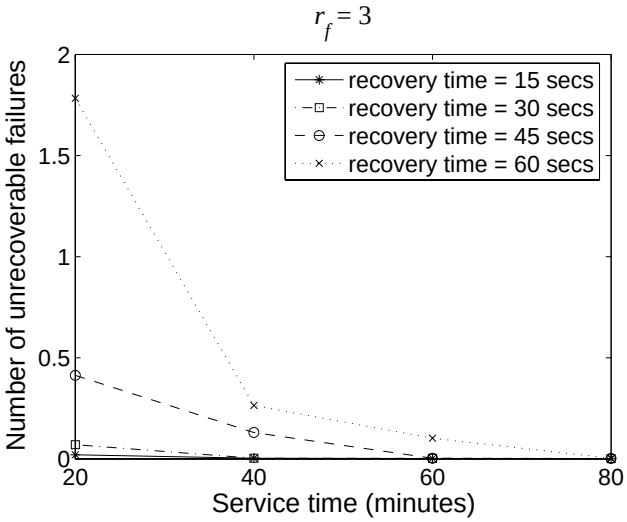


**Figure 29:** Unrecoverable failure, $r_f = 3$

$2 * 10^4$ peers participate the ESM overlay. Each peer departs the system by failing after certain amount of time. The time each peer stays in the system is taken as exponentially distributed random variable with mean $st$, which indicate the service time of a peer in the overlay. It is clear that unrecoverable failure of peers will trigger the re-subscription process and cause the service interruption to its downstream peers. However, we want to observe the behavior of our replication scheme with different $r_f$ values and see how the ESM service in Cascade can be recovered with replica activation, instead of the expensive re-subscription.

The graphs in Figures 27, 28, and 29 plot the total number of unrecoverable failures that have occurred during the whole simulation for different mean service times ($st$), recovery times ($\Delta t_r$), and replication factors ($r_f$). These graphs show that the number of unrecoverable failures is smaller when the replication factor is larger, the recovery time is smaller, and the mean service time is longer. Note that our simulation represents a worst scenario that every peer leaves by failure and no peer enters into the system. However, with a replication factor of 3, the number of unrecoverable failure is negligible.

These experiments show that, although the cost of replication maintenance grows with the increasing replication factor, the dynamic replication provided by Cascade is able to achieve reasonable reliability with moderate values of the replication factor.

#### *2.6.4.2 Service Recovering Overhead*

We measure the overhead of service recovering by the total number of messages exchanged to restore the interrupted service. A unrecoverable failure of a peer causes its downstream peers to re-subscribe, trying to restore the interrupted multicast services by themselves. On the other hand, if a peer's service replica is activated when it fails, only one message is used to report the failure and one fast activation message is involved to activate the service replica.

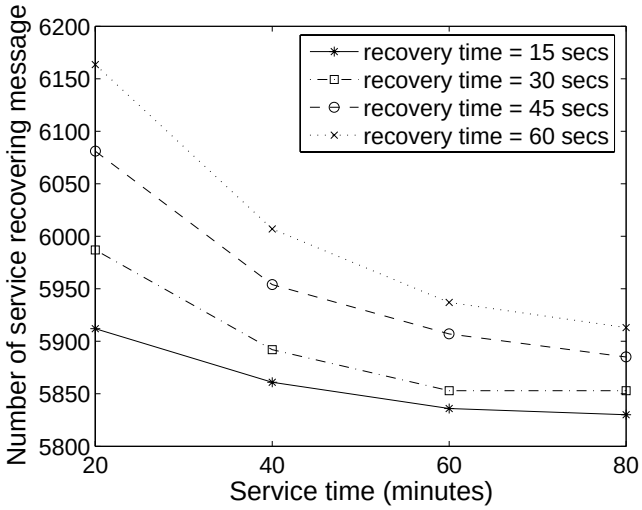


**Figure 30:** Number of service recovering messages under replication scheme, $r_f = 1$

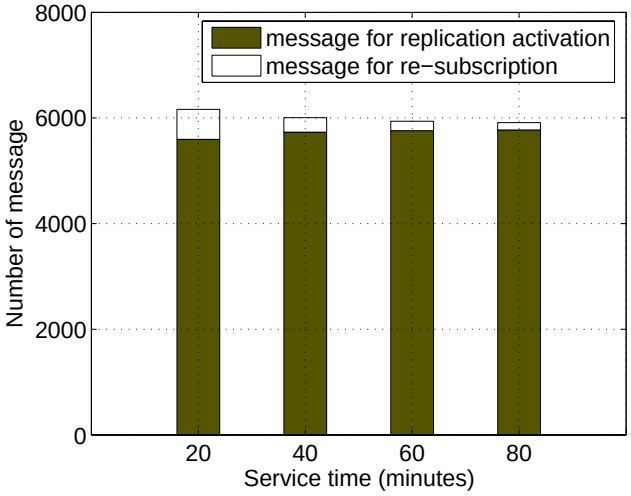


**Figure 31:** Composition of service recovering messages, $r_f = 1$, $\Delta t_r = 60$ sec.

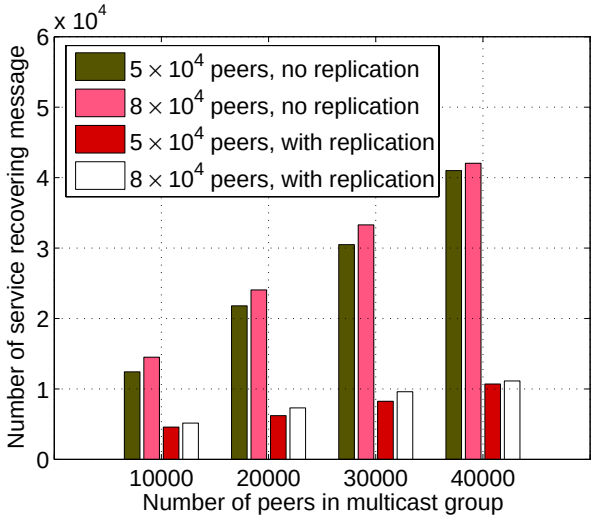


**Figure 32:** Reduced service recovery overhead, $\Delta t_r = 15$ secs, $st = 20$ mins, $r_f = 1$

We observe the number of messages exchanged under the same experiment configurations of Figure 27. We count the total number of messages generated for both replica activation and service re-subscription. The results in Figure 30 conforms to the curves of Figure 27. When the number of unrecoverable failure increases, more messages are generated for the re-subscription requests. However, as plotted in Figure 31, most of the messages are for the replica activation, since most of the interrupted services are restored by the replica activation.

To evaluate the effect of the replication scheme on reducing the service recovery overhead, we compared the number of messages incurred by the replication scheme to the number of messages generated when there is no service replication. We measures multicast groups with $1 * 10^4 \sim 4 * 10^4$ peers built over P2P network with $5 * 10^4$ and $8 * 10^4$ peers. The replication scheme is setup with $r_f = 1$ and the peer service times follow exponential distribution with mean 20 minutes. The experiment results are plotted in Figure 32. The overhead of service recovering increases almost linearly, as the number of peers in the multicast group and the P2P network increases. However, we observe that when the service replication scheme is used, much fewer messages are generated. With the overhead of maintaining ONE service replica, we reduce the messaging overhead by 62.3% to 73.8%.

## 2.7 Related Work

End-system multicast has received considerable research attention in the past few years. Chawathe [18] and Chu et al [20] propose a two-step protocol to build multicast trees. For example, the Narada [20] first generates a mesh network among all members and then uses distance vector to generate the multicast tree. However, because of their high maintenance overhead, such schemes are only suitable for small networks. The Overcast protocol [35] presents a distributed strategy to organize a set of proxies (Overcast nodes) into a distribution tree, with the multicast source as the root. It uses end-to-end measurements to optimize bandwidth among the root and group members. This scheme is in some ways similar to the Yoid scheme [26]. The Cascade system differs from these schemes in that we used completely distributed protocols to construct and maintain the ESM overlay. The efficiency of the Cascade ESM overlay is achieved by leveraging the network proximity information provided by the landmarks signature scheme.

Cascade is similar to ESM protocols like NICE [12], CAN-multicast [46], Bayeux [67], and SCRIBE [17], which use implicit approaches to create ESM overlays. In those systems, a control topology is built by using either the on-shelf P2P protocols like [45, 49, 64] or specially designed protocol like [12].

Nevertheless, those on-shelf P2P protocols used by these works are not specially designed to support ESM. Thus, there is still a large room for performance improvement. The Content Addressable Network (CAN) [45] organizes nodes in a $d$-dimensional Cartesian space, and it uses landmark technique to partition the Cartesian space into various sized bins. However, as Xu et al. [60] demonstrate the overlay structure is constrained by the underlying network topology in topology-aware CAN [47], and the technique used by CAN may destroy the randomness of the nodes distribution, thereby inducing high maintenance costs. The Pastry [49] system exploits topology information in overlay routing using geographic layout, proximity routing, and proximity neighbor selection. It assumes triangle inequality in the network topology and using expanding-ring search to choose the physically closest node at node joining. Scribe [17] is an ESM protocol built on top of Pastry. Our experiments show that our basic Cascade system yields similar performance as that of Scribe.

However, due to *logarithmical deterioration of routing* [60], the stretch of Scribe's multicast forwarding link increases exponentially from the subscriber to the multicast source. On the other hand, the randomness of node identifier distribution of Pastry prevents the usage of optimization techniques such as neighbor lookup of Cascade, which helps Cascade to achieve better utilization of network resources. In Bayeux [67] system the joining request to an existing multicast tree is forwarded to the root node, from where a reverse message is routed back to the new member using the Tapestry [64] P2P protocol. The nodes on this routing path will record the new member's identity and include it into the multicast forwarding path. Similar to NICE [12] system, such a scheme can overload the root node and cause service interruption when the ESM overlay is heavily loaded.

The approach taken by Cascade differs from these existing researches in two aspects. First, the efficiency of Cascade ESM overlay is the result of the synergy of optimization techniques at different system levels. We use the landmark signature technique to improve the routing efficiency at the P2P network level, whereas the neighbor lookup technique optimizes the multicast overlay at the ESM overlay level. Second, Cascade presents a scalable solution for end-system multicast in a heterogeneous environment by using the virtual node scheme. In short, the techniques proposed in this chapter are unique and they comprehensively address the important challenges in end system multicasting.

# CHAPTER III

# INFORMATION DISSEMINATION USING PEER TO PEER GROUP COMMUNICATION

## *3.1 Introduction*

In this chapter, we study the problem of implement large scale information dissemination by using a peer-to-peer group communication system. With the rapid advances of wireless communication technology and the increasing popularity of hand-held devices, we witness the continuous escalation of multi-party group communication applications, such as multiplayer online games, online community based advertising, real-time conferencing [4], and instant messaging [3]. The implementations of these applications usually involve the exchanges of text or multimedia contents among multiple participants. A large number of existing group communication systems achieve scalability through computer cluster-based server farms [4, 3], which are exclusively owned by service providers and require significant administrative investment. Subscribers usually have to pay a premium for scaling the services [4] and are often limited to the features defined by a specific service provider.

Peer-to-Peer (P2P) networks demonstrate a promising paradigm for providing open and distributed information sharing services by harnessing widely distributed, loosely coupled, and inherently unreliable computer nodes (peers) at the edge of the Internet. The success of Skype [6] has shown both the opportunity and the feasibility of utilizing P2P networks as an economical alternative infrastructure for providing wide-area group communication services. In Skype, widely distributed personal computers are interconnected by an unstructured P2P overlay network. By utilizing the high flexibility and low maintenance cost of such an open system architecture, Skype enables value-added services like Internet-to-PSTN (Public Switched Telephone Network) calls at a fraction of the price of traditional telephony services. However, the overlay networks in Skype are typically used *only* for service lookup

and control signaling. Under the multi-party conference settings, the voice communication payloads are directly forwarded to and relayed by a single node through its IP unicast links [13]. As a result, the maximum number of participants allowed in each conference session is limited. The first release of Skype only enables group communication among less than 6 participants [13].

An immediate question to ask is whether a P2P overlay network can be optimized to provide scalable and efficient wide area group communication services. Two types of P2P networks have been extensively studied. Structured P2P networks [45, 49] regulate the network topologies through Distributed Hash Tables (DHT) and provide efficient inter-peer communication in a bounded number of hops. There have been a number of group communication systems built over DHT-based structured P2P networks [17, 46]. However, it is widely recognized that the overhead of maintaining DHT-based structured topologies is significant in a highly dynamic environment and it may cause considerable performance degradation [19]. In contrast, because of their simplicity, Gnutella-like [58] unstructured P2P networks have low maintenance cost against network dynamics, such as peer joining, failure, and departure. Surprisingly, there are currently only limited implementation and deployment of group communication applications over such P2P overlay networks. None, including Skype, to our knowledge, has proposed a scalable group communication protocol for supporting a large number of participants over an unstructured P2P network. The common concern about unstructured overlay networks is their non-deterministic nature in service lookup and their inefficient utilization of the underlying IP network resources.

The design of Peercast attempts to address the following three questions. First, how can we translate wide-area group communication application requirements, such as communication efficiency, load balancing, and system scalability, into the metrics that we can use at the overlay network management layer, and how can we incorporate them into the P2P group communication middleware design such that group communication applications could obtain optimal or near optimal performance? Second, the unstructured P2P overlay networks are randomly constructed. Their tolerance to transient network states is rooted in the fact that they use no global control mechanism to regulate resource distribution and

the network topology. How can we design an overlay network management protocol in such a way that an unstructured P2P overlay network could obtain the features critical to the performance of group communication applications without losing its randomness? Third but not the least, searching or service lookup in Gnutella-like unstructured P2P networks is known to be non-deterministic and network bandwidth-intensive. It may take longer time to locate a service, and a P2P lookup request may generate huge amount of network traffics due to the broadcast nature of the lookup protocol. A number of optimizations have been proposed [19, 5] to improve the lookup efficiency of unstructured P2P networks. However, few of them are designed for scaling wide-area group communication services. An important research challenge in Peercast design is how can we develop an efficient service lookup mechanism that is effective for both control signaling and communication group management?

In this chapter, we present Peercast, a scalable and efficient P2P group communication middleware system based on an unstructured P2P overlay network. In our system, the P2P network serves not only as a signaling overlay, it also carries group communication payloads through distributed spanning trees composed of the unicast links interconnecting multiple participants. Our system has three distinct features.

First, we present a generic utility function for optimizing unstructured P2P network topology and underlying P2P search protocol. Our utility function combines the network proximity and the node capacity metrics for measuring and evaluating the utility of nodes in terms of group communication efficiency and system scalability. We show that different optimization mechanisms can use this generic utility function for peer selection in constructing P2P overlay networks and in building efficient and scalable spanning trees to support group communication applications.

Second, we present a utility-based P2P overlay network management protocol, which uses the proposed generic utility function in constructing unstructured and low-diameter P2P overlay networks. This protocol generates overlay networks that can match structured P2P networks with their high scalability and communication efficiency, while keeping the low maintenance overhead of unstructured P2P networks. A unique feature of this protocol

is that it can balance the per-peer workloads according to the node capacity of each peer, in order to avoid performance degradation and avoid introducing bottlenecks into the system.

The third unique characteristic of the Peercast design is its distributed algorithm for constructing spanning trees that interconnect the participants of the group communication applications. This algorithm is the core of the Peercast wide-area group communication middleware. It enables P2P networks to carry group communication payloads through the unicast links interconnecting participants. In summary, the Peercast communication middleware system offers the following three basic group communication services:

- A service announcement mechanism that selectively propagates the service information to peers in the overlay network. By cutting out the paths that will not likely be used in group communication spanning tree, this mechanism reduces the messaging network traffic by 27% to 56% compared to the popular advertisement scheme used in [18], without affecting the performance of group communication applications.

- A fast service lookup mechanism that enables a participant to discover the services of interest with fewer probing messages. Our service announcement mechanism pushes the group communication information closer to the participants, such that they can locate the service of interest within their overlay network neighborhood much faster, and generate less searching traffic.

- A communication group management mechanism that constructs efficient spanning trees for wide-area group communications. Our experiments indicate that in most of the cases, the end-to-end communication latency between any two peers in the spanning tree is within 3 times of the IP unicast latency. By matching the communication workloads of each peer to its node capacity, we are able to reduce the overloading in the overlay network by one to two orders of magnitudes.

## 3.2 The Utility Function: Combining Node Capacity and Network Proximity

Protocols for constructing overlay networks and organizing wide-area communication groups are typically designed to address two problems, i.e., how should peers choose and maintain their neighbors in the overlay, and how should the connections in the overlay be utilized for group communication applications. Interestingly, we observe that the solutions to both problems address the same design issue - given a list of peers, say $L$, what are the metric(s) that a peer should use to choose a subset of them to connect to.

In Peercast, we address this problem by defining a utility function that assigns different preferences (rankings) to each peer in the list $L$. For any given peer $i$, this function is a weighted combination of two utility metrics: the first metric is for evaluating network proximity, which measures the relative distance between peer $i$ and every other peer in $L$, and the second metric is for evaluating the node capacity of each peer in $L$. These two utility metrics are then combined based on the utility preference of peer $i$, as well as the desired performance properties of the whole overlay.

In order to reach large number of participants or service subscribers, communication payloads in group communication systems have to be relayed within the P2P overlay through spanning trees composed of peers and unicast IP network links, due to the limited access network bandwidth and the limited processing power of each peer. Hence, the properties of the unicast links interconnecting peers in the P2P overlay largely decide the performance and the efficiency of the group communication system. By measuring and utilizing network proximity information, we can optimize both the construction of randomly generated overlay networks and the management of multi-party communication groups in terms of unicast link efficiency. Similarly, it is known that any mismatching between the packet-forwarding workloads and the capacity of peers may introduce bottlenecks in the communication overlay and block the forwarded communication payloads. By measuring and utilizing node capacity information, we can gain better control over the communication workload distribution, in terms of how to utilize the connections in the overlay to provide efficient group communication services.

Concretely, when peer $i$ evaluates a list of peers $L$ to choose a subset of peers, we assume that two types of information are available for each peer $j \in L$: the node capacity $C_j$, and the relative distance between peer $i$ and peer $j$, denoted by $D(i, j)$. We use the accessible network bandwidth to gauge the node capacity of each peer $j$, because the performance of a peer in a distributed environment like P2P networks is largely decided by its access network bandwidth available for forwarding communication payloads. It can be specified by end users as how many 64Kbps connections a peer is willing to support, or be estimated using network probing techniques. We use their network coordinates to estimate the relative distance between peer $j$ and peer $i$. Network coordinates can be measured using mechanisms such as Vivaldi [21] and GNP [2]. For any pair of peers, their network coordinates can be used to estimate the physical network distance between them with satisfying precision.

We define two utility-based preference metrics based on network proximity and node capacity, respectively.

Given a list of peers $L$, we define the $Distance\ Preference$ of peer $i$ to peer $j \in L$ as the probability that peer $i$ chooses peer $j$ out of $L$, based on the network coordinate distance between them. The closer peer $j$ is to peer $i$, the more likely it is chosen. This utility metric will be used to incorporate the network proximity information into the construction of Peercast overlay networks and communication spanning trees. Its definition is given in Formula 4.

$$PD_i(L, j) = \frac{\frac{1}{d_i(L,j)} - \alpha}{\sum_{k \in L} \frac{1}{d_i(L,k)} - \alpha} \tag{4}$$

where $\alpha \in (-\infty, 1)$ is a tunable parameter that indicates $i$'s preference for closer peers. The smaller the value of $\alpha$ is, the less preference gives to close peers. We choose $\alpha < 1$ so that there is nonzero preference on each $j \in L$. The function $d_i(L, j)$ gives the *normalized distance estimation* of $D(i, j)$, which is the network coordinate distance between peer $i$ and peer $j$. We define $d_i(L, j)$ over the list $L$ as follows:

$$d_i(L, j) = \frac{D(i, j)}{\mathrm{MAX}_{k \in L} D(i, k)} \tag{5}$$

After normalization, we have $0 < d_i(L, j) \leq 1$.

Similarly, we define the *Capacity Preference* based utility metric of peer $i$ with respect to peer $j$ as the probability that peer $i$ chooses peer $j$ out of $L$ based on the node capacity of peer $j$. The more resources peer $j$ has, the higher is the probability that peer $j$ could be used to serve more other peers. We will use the value of this utility metric to implement the preferential attachment mechanism for constructing a low-diameter power-law overlay network. It is defined as in Formula 6.

$$PC_i(L, j) = \frac{C_j - \beta}{\sum_{k \in L} C_j - \beta} \tag{6}$$

where $\beta \in (-\infty, 1)$ plays a similar role as $\alpha$ does in Formula 4. $C_j$ is the node capacity of each peer $j$.

When peer $i$ considers selecting peer $j$, peer $i$ should base its choice on its own available resources, or capacity. If peer $i$ possesses more computing power, more access network bandwidth, and more available memory and storage space, we would like to use it as a forwarding hub in the overlay network and applications. For such a peer, it should be connected to those peers that have similar resources and play similar roles in the overlay network, and become a member of the "core" of the overlay network. On the contrary, if the resources of peer $i$ are limited, it should not be placed into the "core" of the overlay network and should avoid playing roles that can easily exhaust its resources. A better choice for such a peer might be connecting to peers that are physically close to it and using them to access the overlay network.

To implement this rationale, we define the *Selection Preference* of peer $i$ to peer $j \in L$ as a utility function of capacity preference and distance preference:

$$P_i(L, j) = \gamma \cdot PC_i(L, j) + (1 - \gamma) \cdot PD_i(L, j) \tag{7}$$

where $0 \leq \gamma \leq 1$.

Choosing different values for parameters $\alpha$, $\beta$, and $\gamma$ gives us the flexibility to fine-tune the selection algorithm for different application scenarios. For an overlay network supporting applications that are sensitive to network proximity, we should set a higher value for $\alpha$ and

a lower value for $\gamma$, such that peers will choose more physical network neighbors. On the contrary, for an overlay network that emphasizes more on load balancing, a higher value for $\beta$ and a higher value for $\gamma$ should be more preferable.

The values of parameter $\alpha$, $\beta$, and $\gamma$ can be mathematically derived by using techniques like the one used in [14], if only we know the exact number of peers and the exact power-law distribution parameters. However, in a distributed environment like P2P networks where global statistical mechanisms are expensive to implement, it is unlikely that such information would be available. In our system, we use the functions defined on the capacity information of each peer to approximate the values of those parameters. We use *Resource Level* $r_i$ to reflect the node capacity of peer $i$. It is defined as the proportion of peers that have less capacity than peer $i$ in the overlay network, and $0 \leq r_i \leq 1$. Specifically, we set the preferential parameters as $\alpha = 1 - r_i$, $\beta = r_i$, and $\gamma = r_i^{-ln(r_i)}$. Such a configuration exactly reflects our design rationale: the capacity of a peer should be used to decide the properties of its connections in the overlay network. More powerful peers should be connected to other peers that are equally powerful and should care less for the distance to their neighbors, whereas peers with limited resources should be connect to peers that are closer to them and avoid being overloaded.

The general utility function for selection preference can then be represented as:

$$P_i(L,j) \quad = \quad r_i^{-\ln(r_i)} \cdot \frac{C_j - r_i}{\sum_{k \in L} C_j - r_i} + (1 - r_i^{-\ln(r_i)}) \cdot \frac{\frac{1}{d_i(L,j)} - (1 - r_i)}{\sum_{k \in L} \frac{1}{d_i(L,k)} - (1 - r_i)} \qquad (8)$$

Note that we use the list $L$ as a sample of the peers in the overlay and estimate $r_i$ as the fraction of peers in $L$ that have less or equal capacity as peer $i$, because no global capacity ranking information is available.

To evaluate the effectiveness of the selection preference metric, we simulate the selecting process of three peers, using a set of synthetic data. We assign each of them with different resource level value. The one with $r_i = 0.05$ represents a peer with low capacity. Similarly, the one with $r_i = 0.5$ simulates a peer with medium capacity, and the one with $r_i = 0.95$ represents a powerful peer. For each of them, we generated a list of $1 \times 10^3$ peers, each of

which is assigned a capacity value that follows a zipf distribution with parameter 2.0. We assume that the distance between each candidate peer and the peer evaluating them follows a uniform distribution Unif(0ms, 400ms).

Figure 33 $\sim$ Figure 38 plot the simulation results, which exactly reflex our design rationale. For a weaker peer that has $r_i = 0.05$, its selection preference to other peers are dominantly decided by its distance to them, as plotted in Figure 33 and Figure 34. On the contrary, the selection preference of a powerful peer is largely decided by the node capacity of peers in the candidate set $L$, as shown in Figure 37 and Figure 38. For the peer that has medium amount of resources, it equally prefers powerful and nearby peers.

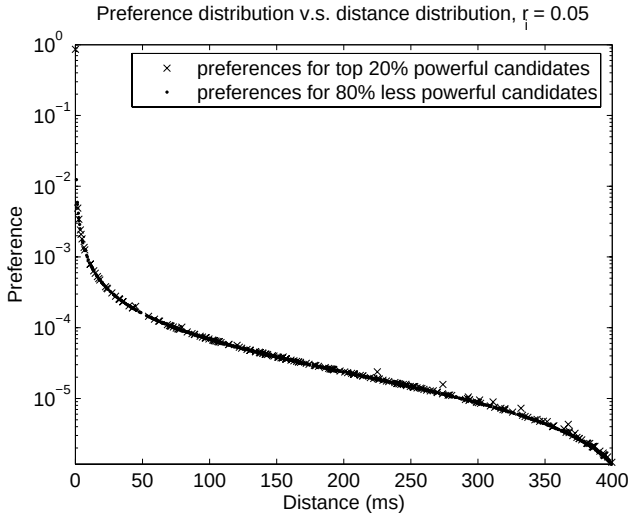


**Figure 33:** Selection preference of low capacity peer vs. distance to other peers

## *3.3 Utility-aware Overlay Bootstrapping*

Research works in natural systems [55] and man-made environments [50, 48] discovered that the topologies of those systems usually follow a power-law distribution. In a power-law graph, the total number of pairs of nodes, $P(h)$, within $h$ hops, is proportional to the number of hops to the power of a constant $\hbar$, $P(h) \propto h^{\hbar}$, where $h \ll \delta$, and $\delta$ denotes the diameter of the graph. The graph can grow while maintaining a low network diameter, i.e., the average shortest path between two peers in term of number of hops.

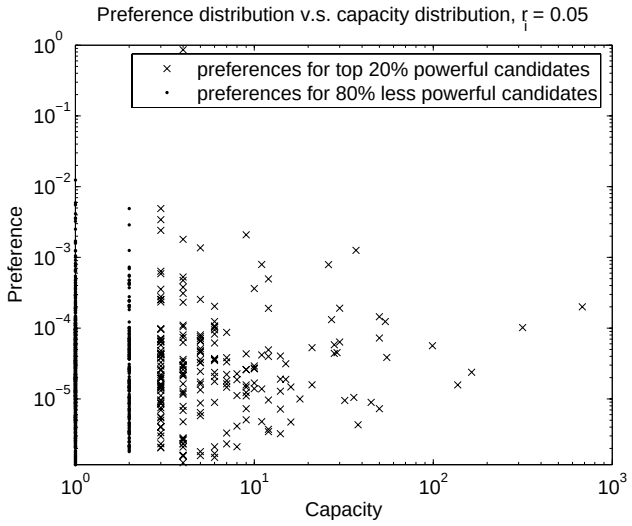


**Figure 34:** Selection preference of low capacity peer vs. capacity of other peers

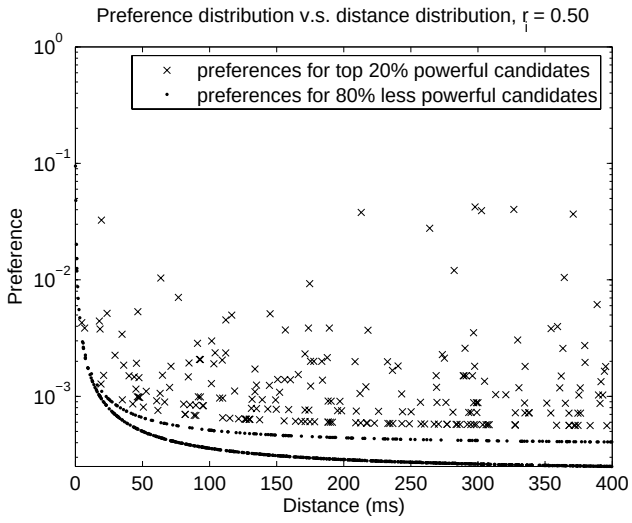


**Figure 35:** Selection preference of medium capacity peer vs. distance to other peers

It has been observed in [48] that the node degree of Gnutella network follows a power-law distribution. However, it is a widely accepted argument [56, 19] that Gnutella P2P networks suffer from large network diameters and searching operations in such networks are expensive. A node needs to either increase the radius of the search scope, or to accept the fact that the query may locate fewer popular objects. Interestedly, we find such believe contradict with the properties of the power-law networks, which should be scale free, as defined in [24].

In the Peercast system, we use a distributed utility-aware algorithm to construct an

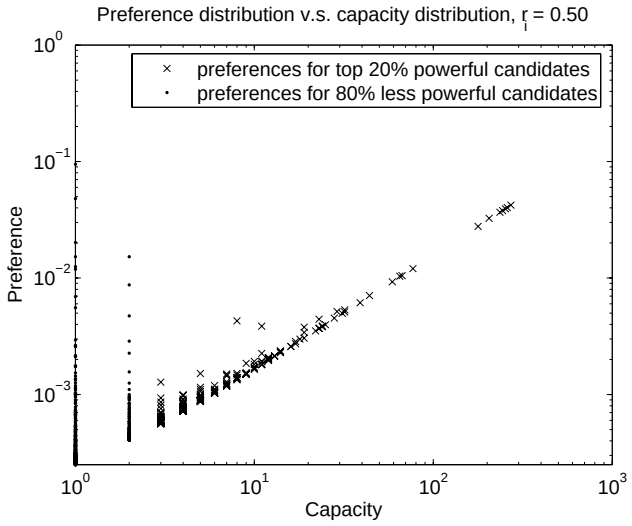


**Figure 36:** Selection preference of medium capacity peer vs. capacity of other peers

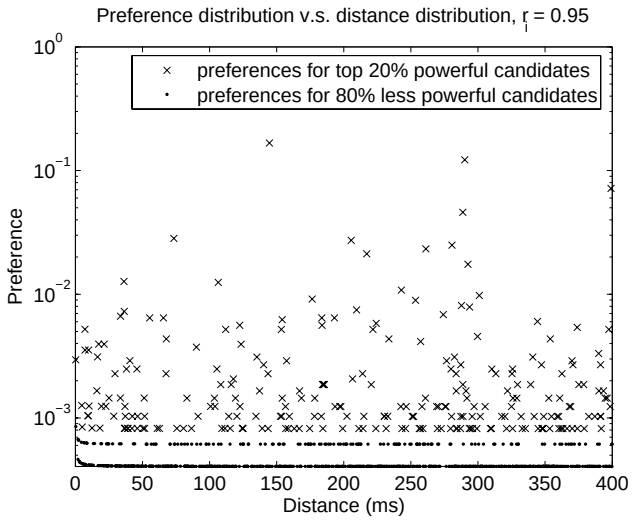


**Figure 37:** Selection preference of high capacity peer vs. distance to other peers

unstructured power-law network. Powerful peers are inserted into the same P2P overlay that other peers participate, rather than being explicitly put into a different routing layer. When a peer joins the overlay, it gathers the information of a number of existing peers as its neighbor candidates. The new peer calculates the probability of connecting to each candidate by using the utility function defined in Formula 8 of Section 3.2. Different weights are put on candidate peers, depending on the capacity of the new peer.

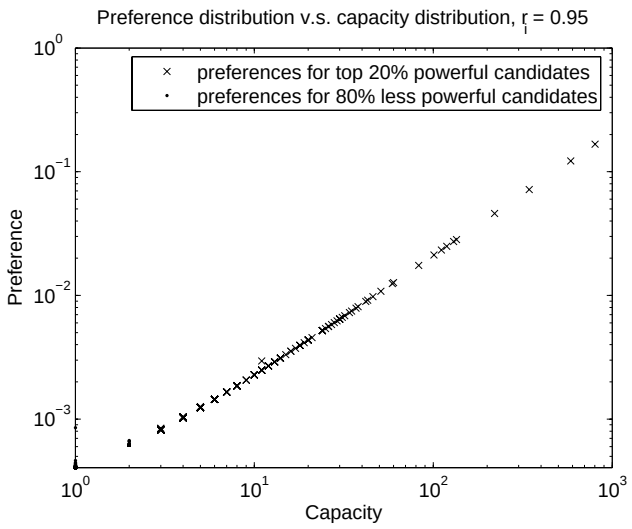


**Figure 38:** Selection preference of high capacity peer vs. capacity of other peers

### 3.3.1 Topology Construction Algorithm

Our protocol extends the current version of the Gnutella protocol [58]. A peer in our overlay is uniquely identified by a tuple of four attributes:

$$< IP\ address,\ port\ number,\ coordinate,\ capacity >$$

where *coordinate* represents its network coordinate, and *capacity* is its capacity.

Preferential attachment has been widely used in centralized network topology generators to generate power-law networks. Algorithms like [14, 42] add links between nodes with probability in direct proportion to the incident link degrees of the nodes. To build a power-law network for wide-area group communication applications, each node needs two types of information to decide its neighbors in the overlay network. First, it needs the degree information of the other nodes to preferentially connect it to those highly connected ones. Second, it needs the network proximity information to connect itself to a set of physical network neighbors and a few randomly selected ones as its routing "shortcuts". However, in a distributed environment like P2P networks, neither kinds of information can be explicitly obtained.

In Peercast, a joining peer $i$ obtains a list of existing peers by contacting a host cache server or using its local cache, which contains its P2P network neighbors carried from the

last session of activities. The host cache server is an extension of Gnucleus [57], which caches the information of a list of peers that are currently active in the P2P network. The joining peer $i$ attaches its 4-tuple identification to the query request sent to the host cache server. Upon receiving a query request from peer $i$, the host cache sorts its cached entries in an ascending order by their network coordinate distances to peer $i$. From the top of this sorted list, the host cache selects a list of peers $BD_i$. They are returned to pere $i$ together with a list of randomly selected peers $BR_i$. We set $|BR_i| = |BD_i|$ and use $B_i$ to denote $BR_i \bigcup BD_i$. We let $5 \leq |B_i| \leq 8$. This is also the default setting used by Gnutella systems [58].

Starting from the subset $B_i$ of bootstrapping peers received upon its entry, Peer $i$ sends a probing message $M_{prob}$ to each peer $k \in B_i$:

$$M_{prob} =_{def} \langle\ source = i,\ type = prob,\ TTL = 0,\ hops = 0\ \rangle$$

Each peer $k$ that receives this probing message sends back a responding message $M_{prob\_resp}$, which is augmented with a list of its P2P network neighbors $N_k$.

$$M_{prob\_resp} =_{def} \langle\ source = k,\ type = prob\_resp,\ TTL = 0,\ hops = 0\rangle$$

Peer $i$ assembles all the neighbor information contained in the probing replies and compiles them into a candidate list $LC_i$. For each unique peer $j \in LC_i$, peer $i$ calculates two types of information: (1) The *occurrence frequency* of peer $j$, which records the number of appearances of peer $j$ in $LC_i$, denoted as $f_i(j)$. As $LC_i$ serves as a sampling of the peers in the network, $f_i(j)$ is the sample of the degree information of each peer $j$. (2) The estimation of the physical network distance between peer $i$ and peer $j$, denoted by $d_i(LC_i, j)$, as defined in Formula 5 of Section 3.2.

Based on its own node capacity, peer $i$ selects a number of peers from the list $LC_i$ and adds them into its neighbor list $N_i$, with probability defined by Formula 9. Note here that we use $f_i(j)$ to substitute the node capacity information $C_j$ and use $LC_i$ to replace the candidate list $L$ of Formula 8. Based on Formula 4 and 6, as well as the settings of $\alpha, \beta, \gamma$ discussed in Section 3.2, The selection preference based utility function can be represented as follows:

$$P_i(LC_i, j) = r_i^{-\ln(r_i)} \cdot \frac{f_i(j) - r_i}{\sum_{k \in LC_i} f_i(j) - r_i} + (1 - r_i^{-\ln(r_i)}) \cdot \frac{\frac{1}{d_i(LC_i,j)} - (1 - r_i)}{\sum_{k \in LC_i} \frac{1}{d_i(LC_i,k)} - (1 - r_i)} \quad (9)$$

The value of resource level $r_i$ of peer $i$ can be obtained in two different ways. The first approach is to use some statistical information like the one presented in Saroiu, *et al.* [50], which gives the bandwidth distribution for a Gnutella P2P network. The second approach is the one we actually adopted. In this approach, the resource level $r_i$ of peer $i$ is approximated by counting the proportion of peers in candidate list $LC_i$ that have equal or less capacity than peer $i$. Such approximation helps avoid the reliance on the statistical information, which may be outdated as the network technologies evolve.

After peer $i$ sets up its outgoing edges (forwarding connections), it will start to setup the incoming edges to itself (back links or backward connections of peer $i$). This task is performed by sending a backward connection request to each peer $k \in N_i$ in the following format.

$$M_{back\_req} =_{def} \langle source = i,\ type = back\_req,\ TTL = 0,\ hops = 0 \rangle$$

The request is augmented with the capacity information $C_i$ of peer $i$ and its network coordinates.

Peer $k$ calculates the probability of setting up a back link to peer $i$ by evaluating the capacity and distance information of peer $i$ against its existing neighbors. The evaluation formulation takes three rankings as inputs and returns a value as the probability that peer $k$ should consider peer $i$ as its neighbor. Specifically, those three rankings are: the capacity ranking $rc_k(N_k)$ of peer $k$ amongst its neighbors $N_k$, which is defined as $rc_k(N_k) = \frac{|\{j|j \in N_k, C_j \leq C_k\}|}{|N_k|}$; the capacity ranking $rc_i(N_k)$ of peer $i$ amongst the neighbor of peer $k$, which is defined as $rc_i(N_k) = \frac{|\{j|j \in N_k, C_j \leq C_i\}|}{|N_k|}$; the distance ranking $rd_i(N_k)$ of peer $i$ amongst $N_k$, which is defined as $rd_i(N_k) = \frac{|\{j|j \in N_k, D(j,k) \geq D(i,k)\}|}{|N_k|}$, where $D(i,k)$ denotes the network coordinate distance between peer $i$ and peer $k$. The probability with which peer $k$ accepts the back connection request is defined as follows:

$$PB_k(N_k, i) \quad = \quad rc_k(N_k)^2 \cdot rc_i(N_k) + (1 - rc_k(N_k)^2) \cdot rd_i(N_k)$$

Such a design reflects the same rationale we followed in devising the peer selection mechanism, i.e., powerful peers are easier to be accepted by other powerful peers as their neighbors, and weaker ones are good candidates only when they are close enough. Peer $k$ generates a random number following uniform distribution Unif(0, 1). If this number is smaller than $PB_K(N_K, i)$, peer $i$ is accepted by peer $k$ as a new neighbor, and a back connection acknowledgement is sent back. If this number is larger than $PB_K(N_K, i)$, only with probability $p_b$, a backward connection from peer $k$ to peer $i$ will be set up. The value of $p_b$ controls the ratio between the number of outgoing links and the number of incoming links of each peer. In our implementation, we set it with a value 0.5.

Parameter $\sigma$ maps the average workload for handling one out-going link to the unit of capacity, its value is decided by the specific applications supported by the overlay network. For applications such as media streaming that heavily consume network bandwidth, $\sigma$ tends to have a larger value. While for systems that support only text message exchanging, $\sigma$ will be given a higher value.

In our implementation, we set the value of parameter $a$ to $u^{2.65}$, where $u$ is a random number following uniform distribution Unif(0, 1). This value is chosen experimentally to give a smaller number of initial neighbors to powerful peers in the overlay network. One reason to do so is to prevent malicious peers from manipulating the topology of a P2P overlay by specifying exaggerated capacities. Also, if we expect that more powerful peers may join the overlay network later, it is important that portion of the capacity of each power peer should be reserved to accommodate the later connections to them.

### 3.3.2 Neighborhood Link Maintenance

Our overlay construction algorithm builds unstructured power-law P2P overlay by guiding each new peer to select its neighbors based on the Peercast utility-aware peer selection mechanism. Once the overlay is constructed, peers will behave like the ones in a normal unstructured P2P network. The simplicity of the Peercast unstructured overlay network

leaves us enough design space to accommodate various optimization techniques like the ones used in [19, 56].

In our system, we use an epoch-based scheme to maintain the structure of the P2P overlay. For any peer $i$ in the P2P overlay, it exchanges heartbeat messages with its neighbors at a certain frequency. With each heartbeat message peer $i$ sends to a neighbor $k$, it attaches its own quadruplet identity (recall the definition in Section 3.3.1). Once a neighbor $k$ receives such a query, it will send back a confirmation message, together with its identity. A departure notification or the consistent timeout in receiving the confirmation message from a neighbor peer $k$ indicates its failure. Peer $i$ will include $k$ into a failure neighbor list $LF_i$. Peer $i$ may accept back-connection requests from other peers. For each back-connection link, peer $i$ will include the adjacent peer of this link into a back-connected neighbor list $LB_i$.

At a certain interval defined by an *epoch* $\Delta T$, peer $i$ tries to repair its neighbor list by establishing links to a number of peers that are currently not its neighbors. The number of these new peers is defined as $MAX(0, |LF_i^{\Delta T}| - |LB_i^{\Delta T}|)$, where $LF_i^{\Delta T}$ and $LB_i^{\Delta T}$ are respectively the failure neighbor list and the back-connected neighbor list gathered in epoch $\Delta T$. New peers are chosen in a similar manner as we use in bootstrapping. The only difference is that we use the neighbor list $N_i$ of peer $i$ to replace $B_i$, and exclude peers in $N_i$ and $LF_i^{\Delta T}$ from the candidate list $LC_i^{\Delta T}$. Both $LF_i^{\Delta T}$ and $LB_i^{\Delta T}$ are refreshed at the end of each epoch $\Delta T$, and the same maintenance process will continue until peer $i$ leaves the overlay network.

The length of epoch $\Delta T$ is dynamically adjusted at each peer, so that the overall overlay network could adapt agilely to network dynamics such as peer joining, departure, and failure. The size of the list $LF_i$ and the size of $LB_i$ indicate the magnitude of change of the neighborhood of peer $i$. However, we need to estimate how fast such change happens, so that we can compensate the loss of its out-going (forwarding) links and incoming links (back links) fast enough such that the services around peer $i$ will not be affected. Specifically, given a series of epochs $\cdots \Delta T_{j-1},\ \Delta T_j,\ \Delta T_{j+1} \cdots$, we have the following relationship among the length of consecutive epoches,

$$\Delta T_{j+1} = (1-\xi)\frac{|LF_i^{\Delta T_{j-1}}|}{|LF_i^{\Delta T_j}|}\Delta T_j + \xi\ \Delta T_{j-1}$$

In Peercast, we set the value of $\xi$ to 0.75 so that the overlay could adjust fast enough to the network dynamics, while avoiding the possible oscillation caused by the fast change of epoch length.

## *3.4 Communication Group Management Mechanisms*

A key component of wide-area group communication applications is the spanning tree that carries the communication payloads. Traditional client/server architecture can be viewed as a special spanning tree of height 1. Such an implementation requires tremendous processing power and network bandwidth at the server side to support a large and possibly growing number of participants. To support wide-area group communication with P2P overlay networks, we design a distributed utility-aware mechanism. For each communication group, this communication group management mechanism builds a spanning tree that connects all the participant peers with selectively chosen links from the overlay, based on the performance requirements of specific group communication applications.

If we model the P2P overlay as a directed graph $G < V,\ E >$, where $V$ represents all the peers and $E$ represents all the links in the overlay, the spanning tree $T_P < P,\ E_S >$ could be defined as a connected, acyclic sub-graph of $G$, where the participant set $P \subseteq V$ and links set $E_S \subseteq E$. The objective of the communication group management mechanisms is to build and maintain the spanning tree $T_P$ for each communication group based on the performance requirements of specific group communication applications.

### 3.4.1 Constructing a Distributed Spanning Tree

Numerous application level multicast or end-system multicast systems have been proposed to solve a similar problem. The spanning tree is called "multicast tree" in those systems. Communication information is usually injected from a single source into the spanning tree and relayed to the other nodes (usually called subscribers) that are the leaves of the multicast tree.

Three categories of systems have been proposed. The first approach is to construct

the tree directly and is represented by NICE [12], Overcast [35], and Yoid [26]. In these systems, subscribers explicitly choose their parents in the spanning tree from the list of candidate peers. Due to the specialization of those protocols, we observe very limited implementation of those systems. The second approach, which is adopted by systems like Narada [20] and Scattercast [18], is to construct the spanning tree in two-steps. A well-connected mesh network is constructed first. And a shortest path spanning tree is created using well-known distributed algorithms in the second step. These systems usually use extensive message exchanging to maintain the quality of the mesh network, which is critical to the performance of the multicast tree. Consequently, the scalability of these systems is limited. The third approach is represented by systems like CAN-multicast [46] and SCRIBE [17]. These systems replaced the mesh network in the second approach with structured P2P networks like CAN [45] and Pastry [49]. The multicast tree is constructed using the deterministic routing interfaces of these P2P networks. Because the P2P network topology is more regulated in these systems, it is much easier to incorporate utility-based information such as network proximity [59, 49, 63] into the overlay network, and thus create more opportunities for optimizing the performance of the multicast group. As we discussed in Section 3.1, DHT-based structured P2P networks are less tolerant to transient peer population and may cause degraded system performance of the applications built upon them.

Our system takes a different approach from the existing ones. Instead of a mesh network or a structured P2P network, we use a Gnutella-like unstructured P2P system to organize participants into an overlay, on which we construct the group communication spanning trees. Leveraging the properties of our unstructured P2P overlay and using selective message forwarding, our communication group management mechanisms can construct a spanning tree with fewer messages, compared to [18]. Our experiments (see Section 3.5) show that the performances of the result spanning trees are comparable to those built using the other three approaches. Moreover, the Gnutella-like unstructured nature of our system makes it easier to develop and easier to adapt to applications such as online conferencing, multimedia stream multicast, or content delivery.

### 3.4.2 Building the Communication Group

The objective of our communication group construction algorithm is to select the edges or the links in the P2P overlay to form the spanning tree that connects all the group participants. The implementation of communication group construction algorithms usually includes the implementation of two functions. First, participants should be aware of the existence of the communication group to which they will join. Second, a newly-joined participant should be able to setup a connection to the existing nodes in the chosen communication group for sending and receiving the communication payloads.

In the literature of end-system multicast, especially in the systems adopting two-step approaches that we discussed earlier, the first task is solved by appointing a node as the rendezvous point or the multicast source, and publishing its information at a well-known location such as a bulletin board system. The other participants will use the identity of this node as the keyword to establish the multicast path, usually by leveraging the search interface provided by the P2P overlay network.

We have identified two schemes for implementing the second function. The first scheme is similar to the DVMRP [22] IP-multicast protocol. Instead of using the IP level network devices such as routers to implement the polling and pruning processes of multicast group management, we use overlay networks and peers. This scheme is adopted by the Scattercast system [18], in which the source node solely advertises route information, and each peer in the P2P overlay forwards this advertisement, and quietly builds the local routing table entries. To remove loops and to avoid the problem of counting-to-infinity, the full path information is embedded into the forwarded advertisement messages. For the purpose of comparison, we refer to this scheme as *Non-Selective Service Announcement* (NSSA) scheme in the rest of this chapter.

The second scheme is adopted by systems like SCRIBE [17]. The multicast source is mapped to a well-known node serving as the rendezvous point. Subscribers use the identifier of the rendezvous point as the keyword in their subscribing requests. The P2P overlay treats subscription requests in the same manner as the routing requests. The regulated system topology and deterministic routing algorithms decide the series of peers that the

subscription request will be forwarded along on the overlay, before the request is received by the rendezvous point or an existing participant in the multicast group. The reverse of this path will be used for forwarding multicast payloads down from the multicast source.

Two properties of our system prevent us from directly reusing those schemes. First, the nature of group communication applications is different from end-system multicast systems. In end-system multicast systems, communication payloads are forwarded in one direction only in most of the cases, down from the multicast source, while in group communication systems, each participant may serve as both the source and the sink of the communication payloads. Second, the unstructured nature of our P2P overlay determines that it can not directly use the reverse searching path as the communication path. Because of the random nature of the overlay topology, searching has to be carried out either by flooding the request or through random walks. The former approach usually causes too many subscription requests being flooded across the overlay, and the latter one may generate search paths that are too long to be used as the forwarding path of communication payloads.

In our system, we proposed a scheme that combines the advantages of these two schemes and avoids their disadvantages. We call our scheme *Selective Service Announcement* (SSA) scheme. In this scheme, the spanning tree for a communication group is established in three steps.

**Step 1: Choosing Rendezvous Point** First, a peer in the P2P overlay is chosen as the rendezvous point. Unlike the rendezvous point in SCRIBE [17], to which all the multicast payloads are first forwarded, our rendezvous point serves as the source of the group advertisement messages and will behave as a normal node in the communication spanning tree. There are several ways to choose such a rendezvous point. It can be setup as a dedicated server donated by a service provider who injects contents into the communication group. Second, for groups that are setup for applications like online conferences, the first participant can initiate a random walk search to locate such a node. The search terminates when it locates a peer that has enough access network bandwidth and computational power to handle the traffic and workloads as a rendezvous point.

**Algorithm 1** Selective Service Announcement

```
1: procedure InitiateAdvertising(Service s) *used by the rendezvous point*
2:     SN_rp ← φ
3:     for m ← 1, R_rp do
4:         rand ← Unif(0,1), j ← 1, sum ← 0
5:         repeat
6:             sum ← sum + P_rp(N_rp, j)
7:             j = j + 1
8:         until rand ≤ sum
9:         SN_rp = SN_rp ∪ {N_rp[j]}
10:    end for
11:    for all k ∈ SN_rp do
12:        call k.advertise(rp, s)
13:    end for
14: end procedure
15: procedure Advertise(Peer source, Service s) *defined for each peer k*
16:    if receivedAdvertising.hasKey(s) then
17:        drop this message
18:    else
19:        receivedAdvertising.add(s)
20:        parent[s] ← source
21:    end if
22:    SN_k ← φ
23:    for m ← 1, R_k do
24:        rand ← Unif(0,1), j ← 1, sum ← 0
25:        repeat
26:            sum ← sum + P_k(N_k, j)
27:            j = j + 1
28:        until rand ≤ sum
29:        SN_k = SN_k ∪ {N_k[j]}
30:    end for
31:    for all i ∈ SN_k do
32:        call i.advertise(k, s)
33:    end for
34: end procedure
```

**Step 2: Advertising** The second phase is for the rendezvous point to advertise the group information to the potential participants of the communication group. We realize that the flooding scheme used in DVMRP [22] and Scattercast [18] will incur redundant messages in the overlay network, especially when the peer population is large and the communication group is relatively small.

Under our SSA scheme, each peer that receives the advertisement message will only select a few of its neighbors to forward the SSA message, rather than flooding the message to all its neighbors. By filtering out the neighbors that will not likely be used in the communication spanning tree, we can reduce the number of messages by as much as 65%, compared to the NSSA scheme. We provide the pseudo-code of our selective service announcement algorithm in Algorithm 1.

In this algorithm, rendezvous point $rp$ first calls its local method *initiateAdvertising* to start the SSA process. The parameter $R_{rp}$ denotes the ranking of the rendezvous point, and is defined as the number of neighbors of $rp$ that have more capacity than $rp$. The utility function $P_{rp}(N_{rp}, j)$ is defined as Formula 8, and will help $rp$ choose the peers either have similar capacities as $rp$ or are physically close to $rp$, depending on the capacity of $rp$. Those peers will be the ones more useful to $rp$ and will likely be included in the spanning tree.

Upon receiving an SSA message, peer $k$ performs two tasks as shown in method *advertise* of Algorithm 1 to forward the advertisement message. First, peer $k$ uses a local hashing table *receivedAdvertising* to check and record if it has already received the same message from any other neighbors. The message will be dropped if it is a duplicated one. Otherwise, the same mechanism we used to initiate the service announcement process on $rp$ will be used to select neighbors of peer $k$ for further propagating the SSA messages.

**Step 3: Subscribe** Subscription activities are initiated when a peer $i$ decides to join a communication group. Two scenarios need to be considered. First, if the potential service subscriber (peer $i$) has already received and routed the service advertisement, peer $i$ is already on the message forwarding path of this communication group. All it need to do

is to start the subscription process by sending the joining message in the reverse direction of incoming SSA message, which is implemented by invoking a call to the *subscribeTo* method of its parent. Second, if the subscriber has never received or forwarded the SSA message before, a search method provided by the P2P overlay is triggered to look up the neighborhood of peer $i$ for a peer who might have received the SSA message of service $s$.

The search method is implemented as a ripple search in standard Gnutella P2P network, with initial TTL (Time to Live) value set to 2. Because our SSA mechanism already pushes the service information close enough to each potential subscriber, a peer can find a nearby neighbor that has received the SSA message with high probability. Our experiment reports that the average success rate of two-hop subscription search is as high as 100%.

Due to the space restriction, we have to skip the topics such as system reliability and security. Please note that we design Peercast as an open platform to integrate security mechanisms like Event Guards [51] and reliability mechanisms like [63]. When replication scheme like [63] is used, our utility-aware peer selection mechanism can identify service replica holders that are either closer in terms of network distance or have higher capacities.

Our system does not exclude the architecture of "supernode" or multi-layer overlay networks. Instead, our scheme can be used to construct the higher layer of overlay that interconnects supernodes. Normal peers can simplify their bootstrapping process by connecting only to supernodes. Our utility-aware peer selection mechanism can help peers to identify the supernodes that are close to them, such that the communication between a normal peer and its supernode could be more efficient.

## *3.5 Experimental Evaluation*

We have implemented a discrete event simulation system to evaluate Peercast. This system is an extended Java version of p-sim [40] system. We used the Transit-Stub graph model from the GT-ITM topology generator [61] to simulate the underlying IP networks. Peers are randomly attached to the stub domain routers and organized into overlay networks using the algorithm presented in Section 3.3. The capacity of peers is based on the distribution gathered in [50], as shown in Table 1. We use the algorithm of [2] to assign network

**Table 1:** Capacity distribution of peers

| Capacity level | Percentage of peers |
|---|---|
| 1x | 20% |
| 10x | 45% |
| 100x | 30% |
| 1000x | 4.9% |
| 10000x | 0.1% |

coordinate to each peer. Each experiment is repeated over 10 IP network topologies. Each IP network supports 10 overlays, and each overlay network has provided services for 10 communication groups.

### 3.5.1 Power-law Overlay and Network Proximity

We first simulated the construction of P2P overlay networks. Peers join with intervals following an exponential distribution Expo(1s). They choose their overlay network neighbors using the utility-aware algorithm described in Section 3.3. Figure 39 plots the log-log degree distribution of a Peercast overlay network of $5 \times 10^3$ peers. It presents a clear power-law distribution. similar to Figure 40, which plots the degree distribution of a power-law network generated using the centralized PLOD algorithm [42]. It is interesting to note that our algorithm cut out the long tails compared to the topology shown in Figure 40. This is because our bootstrapping algorithm and back connection scheme are utility-aware and to some extent are conservative in taking on new peers to replace existing ones. This property results in a lower value of network clustering coefficient compared to the random power-law overlay. Nevertheless, as we will demonstrate in the rest of this section, such kind of overlay topologies can help reduce the messaging overhead without any detriments to the performance of both the overlay and the applications supported by the overlay.

As we discussed in Section 3.1, wide-area group communication applications demand that the topology of overlay network conform to the IP network topology. We compared the overlay networks constructed using our algorithm with the ones randomly generated using centralized PLOD algorithm [42]. For each type of overlay, we simulated the joining processes of $1 \times 10^3$ peers. Figure 41 plots the ones in the Peercast overlay network, and

Figure 42 plots the average distance of each peer to its neighbors in the randomly generated power-law network. It is obvious that in the Peercast overlay network, peers are connected to one another by much shorter unicast links. We observe a few long ones though, which belong to the powerful peers that care less of network proximity in selecting their neighbors. They use those long links to connect to other powerful peers and serve as the forwarding backbone of the overlay network.

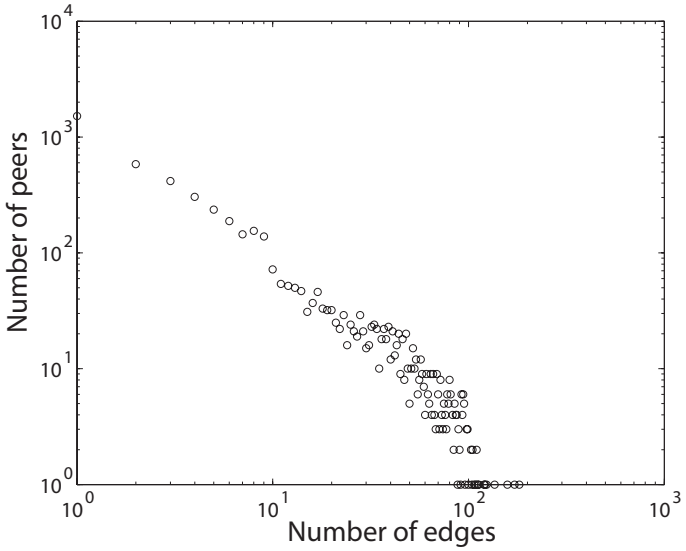


**Figure 39:** Log-log degree distribution of Peercast overlay network with 5000 peers

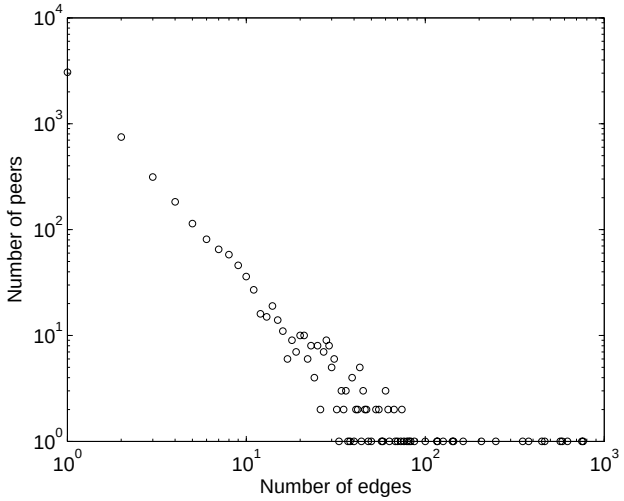


**Figure 40:** Log-log degree distribution of random power-law overlay network, 5000 peers, $\alpha = 1.8$

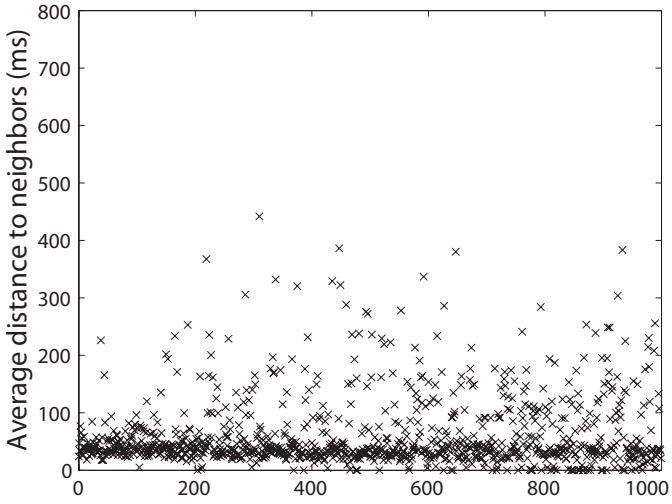


**Figure 41:** Average distance to neighbors of Peercast overlay network with 1000 peers

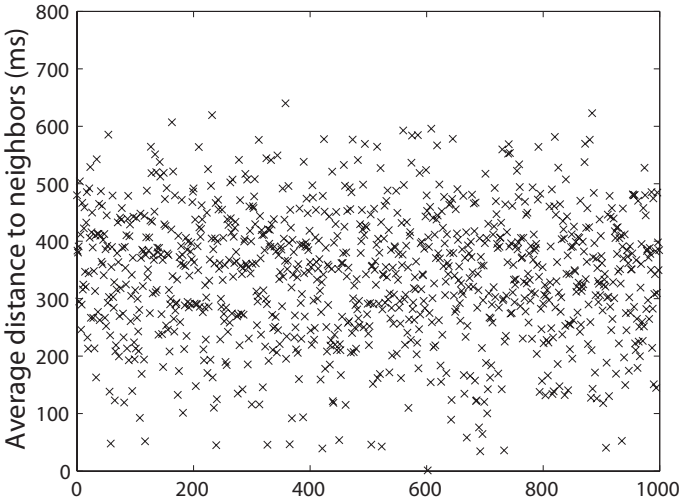


**Figure 42:** Average distance to neighbors of a random power-law overlay network with 1000 peers

### 3.5.2 Service Lookup and Subscription Message Overhead

In most unstructured P2P overlay networks, P2P lookup is implemented using either scoped flooding (broadcast) mechanisms or random walk mechanisms. Flooding-based searches are usually expensive because they require the processing of a large number of messages. In contrast, random walk based searches generate fewer messages but may cause longer query delay. Caching and routing index techniques have been proposed to alleviate this

inefficiency by increasing the availability of the information distributed over the nodes in the overlay and reducing the number of lookup messages. However, those solutions usually assume the knowledge of applications at the overlay network layer and ignore the network proximity properties, which is critical to the performance of group communication applications. In the Peercast design, we have proposed a selective service announcement (SSA) mechanism to improve the messaging efficiency. Compared to the existing systems, our scheme considers both network proximity and capacity of peers when choosing candidate neighbors for propagating and processing service messages.

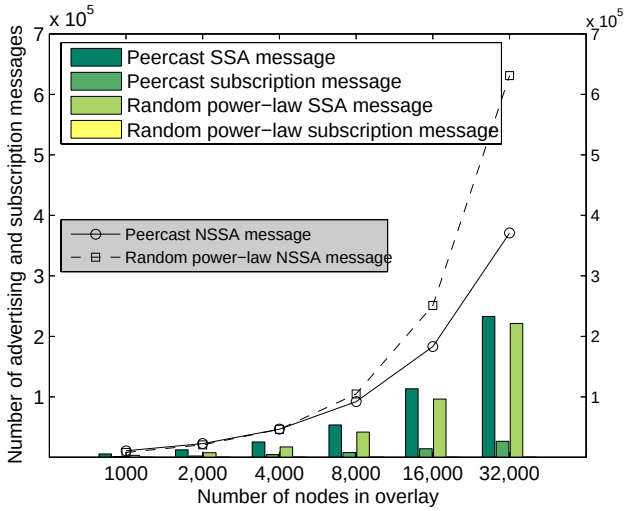


**Figure 43:** Number of messages generated by service lookup schemes

To evaluate the effectiveness and efficiency of the SSA scheme, we simulated the service announcement processes in a number of overlay networks generated using both the Peercast distributed algorithm and the PLOD algorithm. In each overlay network, we randomly select 10 peers as rendezvous points and use each of them to initiate the selective service announcement (SSA) process and non-selective service announcement (NSSA) process (recall Section 3.4).

We first record the fraction of peers that have received the service announcement. When those peers start their subscription process, they can avoid the service searching process because they already know to which neighbor they should forward their subscription requests. Next, we simulate the subscription process of those peers that are not covered by the SSA

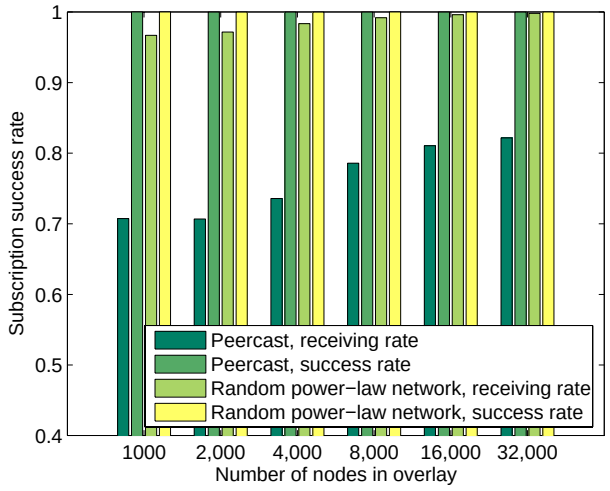


**Figure 44:** Success rate of service lookup in Peercast overlay networks and random power-law overlay networks using selective service announcement

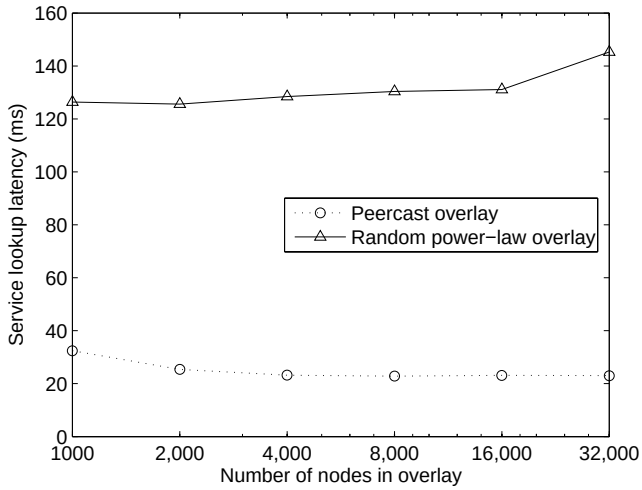


**Figure 45:** Latency of service lookup in Peercast overlay networks and random power-law overlay networks using selective service announcement

messages. They use a ripple flooding search scheme to locate the service of interest, and set the TTL (Time to Live) value of their search messages to 2. We record the success rate of service lookup under both SSA and NSSA schemes in both Peercast overlay and random power-law overlay. Also, we record the total number of messages generated by SSA and NSSA schemes respectively.

Figure 43 shows that the SSA scheme helps reducing the total number of messages generated in both Peercast and random power-law overlay networks. By limiting the number

of messages sent to those neighbors that will not likely be a part of the group communication spanning tree, the total amount of messages of SSA scheme is reduced to 63% to 70% of the NSSA scheme in Peercast overlay network, and 35% to 44% in random power-law overlay. We notice that the number of subscription messages of SSA scheme in random power-law overlays is almost negligible. It is because Peercast overlays have a lower cluster coefficient value than the random power-law topologies generated using PLOD. The SSA messages reach fewer peers. On the other hand, it also shows that the SSA scheme performs better in networks with higher connectivity value.

Figure 44 shows two interesting observations. First, fewer peers in Peercast receive the SSA messages compared to random power-law topology. Second, all subscribers can locate their services with 100% success rate using the subscription message with TTL = 2. This is because by taking into consideration both network proximity and node capacity, the Peercast overlay network provides more candidates of higher utility value, which meet the utility-aware selection criteria when using the SSA scheme. Hence, we notice that the SSA scheme generates more service announcement messages in Peercast than in random power-law overlay networks. However, the peers chosen by our utility-aware selection mechanisms are the ones that are more suitable to the group communication spanning trees and are actually contributing to the success of subscription with a small TTL value.

Now we show how the utility-aware distributed algorithms help reducing the response time of subscription request by taking into account of network proximity in both overlay construction and the SSA forwarding selection. Figure 45 shows that the subscription response time in Peercast overlay network is reduced by 74% to 84%, compared to that in random power-law overlay networks. This property can benefit newly joined peers, since they could subscribe to group communication services much faster in Peercast system than in the random power-law overlay networks.

### 3.5.3 Improvement of Application Performance

We use an end-system multicast system as an example of group communication. End-system multicast has been proposed as an alternative for IP multicast services, due to the

lack of wide acceptance and deployment of IP multicast in the Internet today. In this approach, peers form overlay networks and implement multicast functionality. Multicast data are replicated on peers and propagated over unicast edges of the overlay networks. Compared to IP multicast, end-system multicast systems are less efficient because they may send packets of the same data contents multiple times over the same IP network link. Moreover, the workload distribution among heterogeneous peers affects the overall system performance.

We simulated P2P overlay networks consisting of $1 \times 10^3$ to $3.2 \times 10^4$ peers. P2P overlay networks are constructed using both Peercast mechanisms and PLOD algorithm. We used the routing weights generated by the GT-ITM package to simulate the IP unicast routing. IP multicast systems are simulated by merging the unicast routes into shortest path trees. We use both SSA and NSSA for service announcement and subscription management.

We measured *Relative Delay Penalty* and *Link Stress*, two popular metrics that are usually used to evaluate the efficiency of end-system multicast systems. Relative delay penalty is defined as the ratio between the average end-system multicast delay and the average IP multicast delay. Link stress is defined as the ratio between the number of IP messages generated by an end-system multicast tree and the number of IP messages generated by an IP multicast tree interconnecting the same set of subscribers.

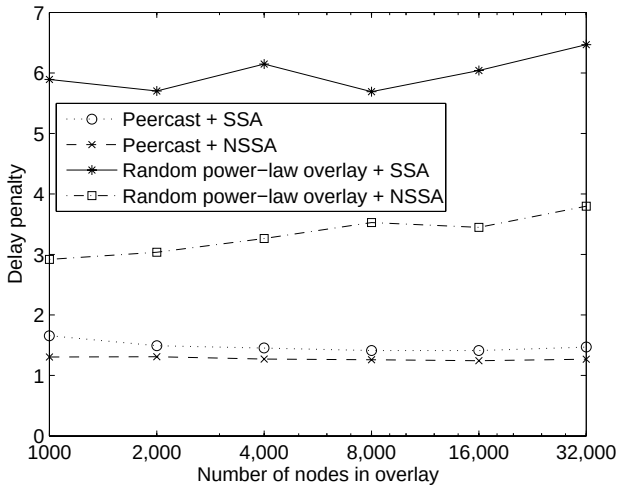


**Figure 46:** Delay penalty of group communication applications

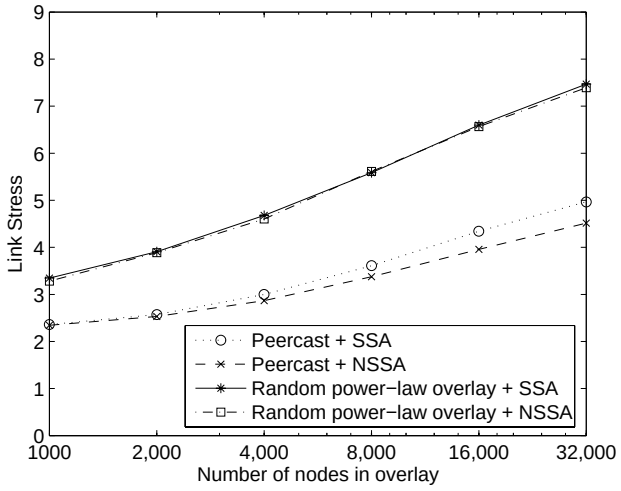


**Figure 47:** Link stress of group communication applications

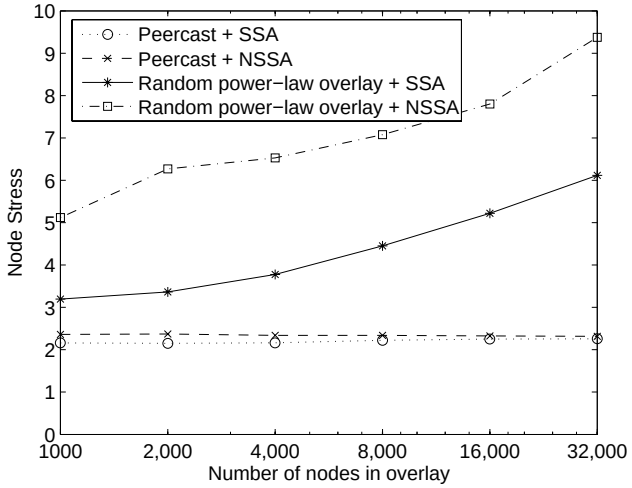


**Figure 48:** Node stress of group communication applications

From Figure 46 and Figure 47, we can see that end-system multicast systems show great performance improvement in both metrics when they are implemented over Peercast overlay networks: the delay penalty is around 1.5, close to the theoretical lower bound of 1 and the link stress is about 2/3 of the ones over random power-law topologies. We attribute such improvements to the fact that Peercast overlay networks successfully incorporate network proximity information into the overlay topology. Multicast payloads are forwarded along much shorter paths (recall the result in Figure 41 and Figure 42), and thus generate fewer IP packets in the underlying IP network.

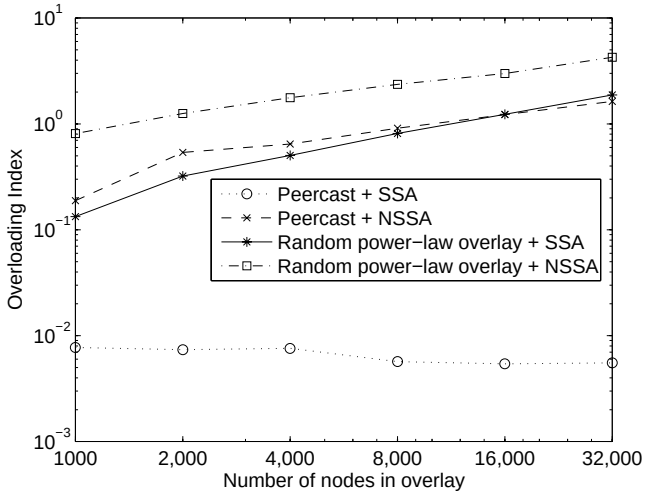


**Figure 49:** Overload index

It is interesting to note that the impact of the SSA scheme on application performance is almost negligible in Peercast overlay networks, whereas the impact in random power-law networks is significant. We attribute this performance enhancement to the fact that Peercast overlay networks are aware of the network proximity of peers, and thus the peers chosen by the SSA scheme are most likely be the ones that are actually used in the information dissemination spanning tree.

### 3.5.4 Improvement of Load Balancing in Group Communication Systems

We compared the impacts of different schemes on load distribution of group communication applications in our simulation. We use a metric *Node Stress* to record the average multicast workloads on peers. It is defined as the average number of children that a non-leaf peer handles in the end-system multicast tree. To measure the load distribution in the overlay network, we define a metric called *Overload Index*. It measures the mismatching between the node capacity and the communication workload. Specifically, we define it as the product of the fraction of peers overloaded and the average workloads that exceed the capacities of those peers.

Figure 48 shows that our utility-aware selection mechanism can improve the load distribution in both random power-law overlay and Peercast overlay. Because the Peercast system considers node capacities in both overlay layer and application layer, it delivers

much better scalability in term of node stress. When the system is scaled to accommodate more subscribers, the node stress in Peercast is almost constant. We also notice that SSA scheme can effectively reduce overloading in overlay networks. By considering node capacity of candidates in the choosing next hop for forwarding service announcement messages, the SSA scheme reduces the overloading in the random power-law overlay by one order of magnitude, and one to two orders of magnitude in Peercast overlay. Combining the SSA with Peercast utility-aware overlay bootstrapping scheme delivers even better performance as we expected. The overloading is reduced by two to three orders of magnitude compared to random power-law network. We also notice in Figure 49 that the line of Peercast overlay using NSSA and the line of random power-law overlay using SSA crossed when the overlay size is around $1.6 \times 10^4$. This phenomenon is what we expected, and indicates that optimization at the overlay level is better than at the application level for larger overlay networks.

## *3.6 Related Works*

Many distributed group communication systems rely on the services of overlay networks for operation [12, 17, 20]. The properties of overlay networks, e.g., communication efficiency, system scalability, and fault resilience, largely decide the performance of those systems. Usually, end-hosts in the communication groups use the unicast links of overlay networks to exchange application and management messages. A communication message may have to traverse the IP network from one end to another multiple times to reach its destination. This is particularly true when group communication peers are widely distributed across the Internet and the overlay topology does not conform well to the underlying IP network topology. Furthermore, nodes in a wide-area network tend to have different computing capacity, different network bandwidth, and different levels of commitment in terms of what and when to share their resources. Such heterogeneity is typically reflected in the different workload amounts they can handle, and the different levels of quality of service that they can provide. Last but not the least, it is widely recognized that wide-area distributed systems like P2P networks are confronted with high turnover rate [50] of dynamic participants.

For example, in both KaZaA and Gnutella, half of the peers participating in the system will be replaced by new peers within one hour. It is critical for the overlay networks used by group communication applications to provide persistent and uncompromised services in such dynamic environments.

Efforts on improving the performance of unstructured P2P networks have proposed mechanisms to optimize the system performance at the application level with the goal of scalable query processing capability. For example, Gia [19] considers only node heterogeneity information in constructing unstructured P2P networks and in processing P2P queries. Featured with a document index replication technique and a specialized random walk searching algorithm, Gia can improve the system capacity in serving file-sharing applications.

A number of P2P systems have been proposed to construct overlay networks with power-law degree distribution because of its scale-free property. Phenix [56] generates power-law topologies of unstructured P2P networks using a distributed version of preferential attachment mechanism. Pandurangan, *et al.* [43] assume a stochastic arrival and departure pattern of peers. They trace the status of peers and coordinate the connections among them with a central server, which limits the system scalability and reliability. In contrast, our system uses only a lightweight bootstrap server, which caches only the partial knowledge of the P2P network and is involved only when a peer joins the overlay network.

However, none of these works has considered combining node capacity and network proximity metrics in constructing overlay networks and the effects such a combination may have on the overlay network performance. We claim that the performance gain through a careful combination of node capacity and network proximity metrics can be significant for wide-area group communication applications.

Another approach to improving P2P networks is to utilize the ranking of different peers in terms of their node capacity and organize them into different overlay service layers. For unstructured P2P networks, KaZaA [5] uses the notion of "supernode" and Gnutella v.0.6 [58] uses a similar notion of "ultrapeer". In structured P2P network, such peers are referred to as "supernodes" and are organized into another layer of overlay called an "expressway" [59] to accelerate the routing services. Generally, those powerful peers are

assigned with heavier workloads and serve as the "forwarding hubs" in the overlay network. Ordinary peers are explicitly connected to them. Such predetermined hierarchical structure introduces a number of system vulnerabilities. First, supernodes are assumed to be stable and possess enough resources. When they are attacked or overloaded, the overlay network might be fragmented if normal peers rely solely on a few supernodes for services. Second, to efficiently route the requests from normal peers, each supernode keeps the state information of the normal peers served by it. However, such state information is usually closely tied to the application, and it is hard to design a supernode overlay layer that can serve as a generic middleware to support different services. Finally, such a system would be vulnerable when malicious peers assume the role of supernodes and trick other peers into relying on them for services.

Adaptation mechanisms have been studied in a few pieces of work [11, 66]. Through successive and periodical refinements of the initial spanning tree for application layer multicast or data streaming services, the tree is incrementally transformed to a more cost-effective one. Our Peercast system can provide complementary features for those systems. They can use our protocols to construct well-regulated spanning trees out of a large number of subscribers as the starting point for future refinements. Our protocol can help reduce the number of adaptations by ensuring the initial efficiency of the spanning trees, and reduce the service interruptions caused by the topology transformation on the spanning trees.

Research works such as RON [10] have been designed to build generic overlays independent of the applications built on top of them. Optimization techniques such as [19] can be used to improve the performance of the overlay networks at the application level. The Peercast development differs from those works in a number of ways. First, our system distinguishes the distance of peers and construct overlay networks that incorporate network proximity information. Second, our algorithm builds "scale-free" power-law topologies and assigns peers with different peer connections according to their capacity.

Compared to DHT-based structured P2P systems [49, 45, 52] and their optimizations [59, 65], our system is more resilient to network dynamic and is easier to implement. Our algorithm is fully distributed and based on only local information. It makes no assumptions on

the underlying IP network and the knowledge of the peer activity pattern.

# CHAPTER IV

# INFORMATION DISSEMINATION USING GEOGRID: A GEOGRAPHICAL SERVICE NETWORK

## *4.1 Introduction*

In this chapter, we study the problem of disseminating geographical information using GeoGrid, a decentralized service network. The rapid growth of wireless communication technologies enables the information dissemination to hand-held devices, which are increasingly gaining popularity. A substantial number of them have multimedia capabilities. Features such as email and text messaging are wildly available on a variety of person electronic devices. Currently available applications include video and audio streaming, event notification, wireless instant messaging, GPS navigation service, and location-based advertisement, just name a few. Such applications are expected to serve end users' requests like "Send me the traffic camera images taken at 10th and I-75/85 in the next 30 minutes", "Tell me the locations of three gas stations within 5 minutes driving and offering #87 gasoline at a price lower than $2.85 per gallon", "Is there any vacant parking space around the Georgia Dome?", and "Are there any of my friends in Buckhead right now?".

The problem we are trying to solve in this chapter is how to build a generic and efficient P2P service network to support such applications. Our solution is based on a few basic assumptions. First, we assume that there exist information sources that can provide the geographical contents requested by the end users. In the examples given above, such information sources could be the traffic monitoring cameras, the owners of gas stations and parking lots, and the people who are willing to share their current location information. Second, we assume that the people ask for information from our service network can be from outside of the network or be inside the network. By "inside of" the network, we mean that they can join the network as peers and handle queries of other users. By "outside

of" the network, we mean that they can be just the consumers of the information, without sharing any of their resources. Our third assumption is about the network nodes in the service network. To simplify our design, we assume that they are not mobile. Compared to mobile devices, desktop computers usually have more access network bandwidth, more stable connections, and more storage space. The latest survey [1] shows that more than 60% of American families have at least one personal computer, and more than 54% of American families have Internet accesses. Furthermore, advances in wireless communication technologies have enabled the point-to-point TCP/IP communication between mobile devices and fix nodes. Peer-to-Peer (P2P) network technologies show a promising paradigm for harnessing distributed resources for information processing and dissemination. It is a nature extension to P2P overlay networks to use the abundant processing power and storage capacities of end-systems to process and cache information for mobile devices.

Specifically, our system is designed to address three outstanding challenges. First, how should we design a scalable network topology such that end nodes tagged with geographical information can be effectively organized into an efficient service network? How can we design such a system that end-to-end communication between any two end-system nodes is bounded? Second, how should we design the object management protocol of the service network such that the objects tagged with geographical information can be efficiently managed by the service network? How should we design the query routing protocol of the service network such that the information consumption requests like object queries and information subscriptions can be handled efficiently? Third, the location-based information usually shows certain spatial and temporal clustering patterns. For example, the highway system in a metropolitan area is usually heavily loaded during the rush hours. In the morning, the highways leading in town are usually crowded, while the out-town routes are heavily loaded in the afternoon. During a sport event, the parking lots around the stadium are usually full, whereas in the most days of the week, those parking lots are sparsely used. To support queries on those information using heterogeneous end-systems, the service network should be able to handle such workload imbalance gracefully, minimizing the possible service interruption and information losing.

Several categories of solutions have been proposed to support the applications we described earlier. The first approach relies on centralized servers to store, process, and disseminate geographical information. Usually, servers are the bottleneck of the system and usually have the problem of scalability. Tremendous efforts like [30] are needed to shed the workload off the server. However, relying on central server or server farms implies that service providers have the total control over the service features available to the end-users. Service subscribers have to pay a premium to the service providers for using their services. Without the supports from the open source community, the deployments of new applications and features are controlled by the service providers. Because of the profitability pressure, they are less motivated to provide services that have relatively small subscriber population. The second approach uses unstructured Peer-to-Peer (P2P) network. Examples include Rebeca [41], Siena [16], and the PeerCast system that we present in Chapter 3. Because of the lack of structure information in the P2P network, end user queries and subscription requests are usually handled in a flooding manner. Semantic information of applications is usually used to control the flooding of the queries or the subscription requests. Deprived of the application semantic information, the messaging overhead in those systems will be considerably more expensive. Compared to these systems, GeoGrid is designed to support generic geographical information dissemination and sharing, and consequentially can not use their solutions. The third approach is using structured P2P network like P-Grid [7], Chord [52], Pastry [49], and Tapestry [64] to support these applications. Solutions proposed include end-system multicast systems like SCRIBE [17] and Buyeux [67]. Because the identifier space of these P2P network are of one dimension, mapping mechanisms like the space curve filling techniques used in [60] and [28] are required to map the objects from multi-dimensional space to the one-dimensional P2P identifier space for efficient processing. While the queries on one object can be handled deterministically and efficiently in those systems, multi-dimensional range queries like "Tell me the locations of three gas stations within 5 minutes driving and offering #87 gasoline at a price lower than $2.85 per gallon" may not be handled elegantly in those systems. The reason is that the space curve filling

technique may not be able to preserve the geographical proximity of objects in the one-dimensional P2P identifier space. Extra efforts are required to preserve such information and will complicate the system design.

In this chapter, we propose GeoGrid, a service network management system that enables geographical information dissemination among wired and wireless end-users. Our proposed solution includes three parts that distinguish it from existing systems. First, we propose a mechanism to organize end-systems into a generic service network. The 2-dimensional geographical location of users and their mobile devices are used to organize end-systems and regulate the overlay topology, which is directly mapped into a 2-dimensional identifier space. By implementing the communication interfaces exported by the service network, various applications require requiring geographical information dissemination services could be supported. Second, we propose a dynamic load balancing scheme for GeoGrid service network. Through local adjustment on end-system node distribution, this scheme dynamically adjusts distribution of information storage, processing, and dissemination workloads, alleviates overloading in the service network, and improves application quality of service. Third but not the least, we design a set of mechanisms for improving messaging routing efficiency and routing workload balance. Messages in the GeoGrid service network are routed based on the geographical location of participant peers and the contents of the disseminated information. By exploiting the geographical proximity of peers and distribution of events, we can significantly reduce the communication overhead in terms of the total number of messages generated and the total amount of network bandwidth consumed. Utilizing the randomized routing shortcuts, our service network can achieve routing latency of $O(\log N)$, which measures the average number of hops a message is forwarded to answer a routing request.

In the rest part of this chapter, we first describe the basic GeoGrid design in Section 4.2 and give a few application examples in Section 4.3. Section 4.4 presents the load-balancing mechanisms of GeoGrid and Section 4.5 presents the enhanced routing scheme. We present the experimental results in Section 4.6. Discussions on related works are in Section 4.7.

## 4.2 Basic GeoGrid System Design

In general, a GeoGrid system is a P2P network interconnecting end-system nodes tagged with geographical information. An end user uses this service network by connecting her mobile devices to one of the nodes in the network, usually through wireless or wired network connection. Each node runs the GeoGrid communication middleware and serves as the proxy of end users. Depending on the applications implemented over the GeoGrid middleware, a proxy can submit queries, process data, and cache query results and event notifications on behave of end users. A proxy can be a personal computer or a server running by a service provider, as long as it can communicate with the other GeoGrid nodes and transfer data onto the mobile devices of the end users that it represents.

### 4.2.1 System Components

A GeoGrid P2P network can be visualized as a plane in a 2-dimensional space. Such a plane usually represents a geographical area, which could be a metropolitan area, a state, a country, or a continent. A GeoGrid plane is usually partitioned into a set of regions, each of which represents a rectangular area in the geographical plane. A *region* $r$ is denoted as a tuple of 4 attributes: $<x,\ y,\ width,\ height>$, where $(x,\ y)$ represents the longitude and the latitude coordinate of the southwest corner of $r$, and $(width,\ height)$ represents the dimension of the region. Two regions are neighbors when their intersection is a line segment. We say that a point $(x,\ y)$ is *covered* by a region $r$ when the following relationship is satisfied:

$$(r.x < o.x \leq r.x + r.width) \wedge (r.y < o.y \leq r.y + r.height) \tag{10}$$

A *node* $p$ is identified by a tuple of 5 attributes: $<x,\ y,\ IP\ ,\ port,\ properties>$. $(x,\ y)$ represents the geographical coordinate of node $p$. $(IP,\ port)$ is the IP address and port number that this node uses to execute GeoGrid middleware. And *properties* represent application specific information such as *capacity*, which quantifies the amount of resources that node $p$ is willing to dedicate for serving other nodes. For different applications, capacity of a node may have different meanings. It may represent the available storage space for

file sharing services, and may represent the available uploading network bandwidth for multimedia streaming applications. In GeoGrid, we use it to represent the available network bandwidth of a node. We consider network bandwidth as the bottleneck that limits the performance of an end system because the advances in computer hardware technologies have alleviated the limitation on the storage and computing capacities of end-systems.

An *object o* in the GeoGrid is represented by a triplet <*x, y, content*>. Depending on the applications supported by the GeoGrid, the content of an object may have different semantics and values. It could be the availability of a parking lots, the gasoline prices offered by a gas station, or a piece of for-rent advertisement. We say that an object *o belongs* to a region $r$ when its coordinate is covered by $r$. In the basic design of GeoGrid, all the objects belong to a region is stored on a node that owns this region. This node will handle all the operations on these objects. Examples include posting of new objects, removing of old ones, and querying and updating on existing ones. The semantic of object management is defined by the applications supported by the GeoGrid.

### 4.2.2 Routing in GeoGrid

The basic GeoGrid system implements one simple operation: given a routing request identified by a coordinate ($x$, $y$), a GeoGrid will return the information of the node that owns the region that covers the coordinate. The routing request is forwarded from its initiator within the 2-dimensional space of GeoGrid. At each step, the owner node of a region handles the request by forwarding it to another that is the closest to ($x$, $y$) among its immediate neighbors. The forwarding terminates when the request reaches the region that covers ($x$, $y$). Figure 50 visualizes a GeoGrid system with 15 nodes. A routing request is initiated by region 13 for a point covered by region 5. The requested is forwarded through region 10, 11, 6, and 7.

Routing between a pair of randomly chosen regions is of overhead $O(2\sqrt{N})$ in terms of number of routing hops. To see how such an overhead is estimated, let first assume that all $N$ regions are of uniform size and are distributed uniformly in the 2-dimensional space. On average, along each edge of the geographical plane, there are $\sqrt{N}$ regions. At each step,

a routing message will be forwarded either along the latitude or the longitude direction. In each direction, it will pass on average $\sqrt{N}$ regions. Thus, the total number of routing hops for a request is of $O(2\sqrt{N})$ hops. When the region sizes are not uniform or the region distribution is not uniform, the average routing overhead may be higher than $O(2\sqrt{N})$.

### 4.2.3 Bootstrapping Process of GeoGrid

To enable geographical routing, each node in GeoGrid maintains a *neighbor list*, which contains a list of nodes that own its neighbor regions. The basic GeoGrid bootstrapping and maintenance process is designed to maintain the integrity of the neighbor list of each node.

The basic bootstrapping process of a node $p$ includes three steps. First, Node $p$ obtains its geographical coordinate by using services like GeoLIM [33] of GPS (Global Positioning System). Second, Node $p$ obtains a list of existing nodes in the GeoGrid service network through a bootstrapping server or from a local host cache carried from its last session of activity. In the third step, Node $p$ initiates a joining request by contacting an entry node randomly selected from this list. The joining request is routed to the region that covers the coordinate of the new node, in a manner similar to the routing of a query request. The new node $p$ splits this region with the current owner node of the region by copying its neighbor list. Node $p$ will purge the neighbor list entries pointing to regions that are not adjacent to its region. The splitting of a region is in latitude and longitude directions alternately. The node whose region is split notifies the peers in its neighbor list of the joining of the new node. Node departure is handled by having the owner of one of the neighboring regions to take over the region owned by the departing node.

### 4.2.4 Discussion

We have two sets of messages exchanged in a GeoGrid. One set of messages are for the management of GeoGrid service network, and includes messages for splitting and merging region, heart-beat, request routing, load-balancing, routing table maintenance, and randomization of routing entries. The syntax of these messages is defined by the GeoGrid middleware, and the exchanging of these messages is transparent to the applications running over GeoGrid

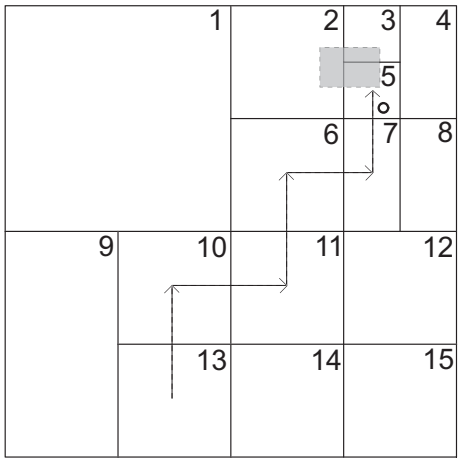


**Figure 50:** Basic GeoGrid service network

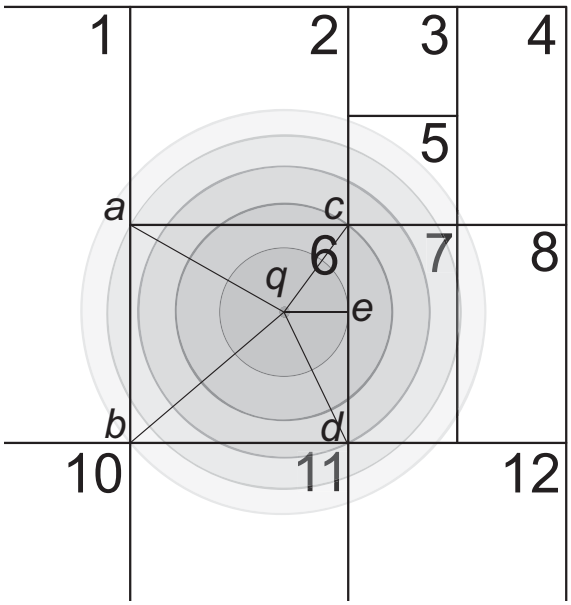


**Figure 51:** Nearest neighbor query using GeoGrid service network

service network. The other set of messages are for applications supported by GeoGrid middleware. Whereas the applications are free to define the syntax of these messages, we require the messages routed using GeoGrid middleware to supply the geographical coordinates of their destination points. GeoGrid middleware handles only the routing of these messages and leaves the handling of application-specific objects and messages to the implementation of these applications, in order to keep the design of GeoGrid as generic as possible.

In the next section, we will first show how the basic GeoGrid system is capable of providing generic routing and searching functions for geographical information dissemination. Nevertheless, there are a few aspects of the basic GeoGrid system that we can improve for providing better services to the applications supported by GeoGrid. The first aspect is the lack of a load balance control mechanism in basic GeoGrid system. The size and location distribution of regions is decided by the joining timing sequence and the location of nodes in the GeoGrid network. When the system demonstrates non-uniform distributions in terms of node locations, node capacities, and workload assignment, we need an effective load balancing scheme . In Section 4.4, we propose a set of local adjustment scheme for dynamically adjusting distribution of workload and alleviating overloading in the overlay. The second aspect of basic GeoGrid system that we can improve is for the routing in GeoGrid. The number of forwarding hops per routing is of $O(2\sqrt{N})$. Messages in the GeoGrid network are routed based on the geographical location of participant peers and the contents of the disseminated information. By exploiting the geographical proximity of peers and distribution of events, we can significantly reduce the total number of messages generated and the total amount of network bandwidth consumed. Utilizing the randomized routing shortcuts, our service network can achieve routing latency of $O(\log N)$, the average number of hops a message is forwarded to answer a routing request. In Section 4.5, we will introduce this shortcut technique.

## *4.3 Application Examples*

Using the routing interface exported by the GeoGrid middleware, a spectrum of applications can be implemented. Although implementations of these applications are not the focus of this chapter, we use a few of them as examples to demonstrate the flexibility of GeoGrid in supporting applications that require geographical information.

**Publisher/Subscriber System** Such a system could be used to handle requests like "please inform me of the traffic around Exit 89 on I-85 in the next 30 minutes". Studies

represented by Siena [16, 15] focus their efforts on the processing of advertising, subscription, and publication messages. It will not be a trivial task to support location-based event processing and routing in those systems because the overlay networks used by them are not aware of the geographical information, and those systems will have to periodically broadcast service announcement messages to inform the subscribers of the existences and locations of publishers and services. In GeoGrid, end users submit their interest on events as subscription requests. Each subscription request $q$ is identified by a rectangular geographical area $< x,\ y,\ width,\ height >$ in which the user intends to query objects or events. A subscription request could be handled in two steps, in each of them the messages are forwarded in a deterministic manner. First, the request is routed toward the region that covers the center of the request, which is a point with coordinate $(x + width/2,\ y + height/2)$. In the example given by Figure 50, we use the gray rectangular to represent the area covered by a subscription request. Its center is covered by region 5. Correspondingly, the subscription request is forwarded first to the node that owns region 5. From there, the owner node of region 5 exams its neighbors to see if any of them intersect with the subscription request area and forwards the request to them. In our example, region 2 and region 3 will receive the subscription request.

The reverse routing path of a subscription request will be used for disseminating events to the subscriber. Each event in GeoGrid is tagged with the coordinate of the location at which the event is captured. An event is routed first to the proxy node that owns the region covering its location. From there, the information is forwarded to all its subscribers, filtered on the path according to the evaluating criteria defined by the subscription requests. In our example, once the owner node of region 2 is notified by an external source of the traffic condition in its region, it will evaluate all the subscriptions it received. Suppose this event happens in the area covered by the gray subscription rectangular. Region 2 will first forward this event to region 5, who covers the center of this subscription request. From there, the event is routed back to the subscriber node and is cached there for the end users to retrieve.

**Nearest Neighbor Query** Such applications are used to answer location-based queries like "tell me the addresses of three gas stations within five miles offering #87 gasoline at a price lower than $2.86 per gallon". Typical solutions [30] use central servers that store all the object information from a certain area. A GeoGrid system can answer such queries in a distributed manner. Information like locations of gas stations and the prices of gasoline can be stored in the GeoGrid P2P network, indexed by their geographical locations. An end user submits a nearest neighbor query tagged with her own location. The request is routed to the node that owns the region where the user is in. If this node has enough information to answer the query, it will forward the result back to the proxy node of the user. Otherwise, depending on the radius of the query, this node may forward the query request to a number of its neighbors.

Figure 51 gives such an example. An end user at location $q$ is asking for the nearest neighbors. If the radius of the request is less than the length of line segment $qe$, the owner node of 6 can answer this query. If the radius of query exceeds the coverage of region 6, like $qc$ does, the request will be forwarded to neighbor regions such as region 2 and region 7. If the end user does not specify any radius of the query, the proxy node will evaluate this query in a multi-step heuristic manner. Starting with the max radius covered by the region owned by the proxy node, the query is augmented with stepwise increasing radius until sufficient results are returned. In our example, node 6 increases the radius of a generic query request following the sequence of $qe$, $qc$, $qd$, $qa$, $qb$ and so on, until the query is fully answered or a messaging upper limit has been reaches.

**Application Layer Multicast** Application layer multicast systems [12, 11, 17, 20] have been proposed as the practical replacement of the IP multicast systems [22, 23]. In application layer multicast systems, the unicast links of end-system nodes are used to connect them into a spanning tree and to carry multicast payloads. Location-based versions of application layer multicast systems can be used to serve request like "send me the snapshots taken by the monitoring cameras at the intersection of I-85 and I-285 in the next 50 minutes", and "send me the live video of July 4th parade on the Peachtree street".

GeoGrid system can serve such applications in a similar manner as it serves the publisher/subscriber applications. A multicast subscription request is tagged with the location information of the multicast source. Nodes in GeoGrid forward the subscription requests toward the multicast source in a similar manner as they handle GeoGrid P2P routing requests. The forwarding of a subscription request terminates when the request reaches a node that is already a member of the spanning tree, or when it arrives at the multicast source. All the nodes on the forwarding path of a subscription request form a multicast path that connects the subscriber node into the spanning tree. When new information is available, it will be disseminated along the multicast tree towards all the subscribers. We multiplex the example given by Figure 50 to illustrate how such a system works. A node in region 13 wants to receive the multicast information from the source node that owns region 5. It will initiate a subscription request that will be forwarded towards region 5. Application multicast contents will be forwarded to the subscribers through nodes that own region 7, 6, 11, and region 10 in sequence. Please note that the proxy node in region 13 can cache and convert the multicast video stream for mobile end-users, even though the wireless link of their mobile devices may be intermittent or their screen resolutions may not match the resolution of the video stream.

## *4.4 Dynamic Load Balancing in GeoGrid*

In a real system, we can not expect that all the objects and events handled by GeoGrid to follow a geographically uniform distribution. The nature of location-based information dissemination implies that some hot spots may exist at different times and at different locations in the service network. The traffic monitoring application gives such an example. In a metropolitan area, the traffic congestions are likely to be in the downtown area in the morning. In the afternoon, the traffic congestions are likely to be on the road segments and intersections leading to out of town. When end users use a service network like GeoGrid to trace the traffic data for minimizing their travel time, network traffics, and information processing workloads will follow a similar distribution pattern in the service network. Due to the heterogeneous nature of end-system nodes, some of them may be overloaded and

become the system bottlenecks.

The problem of load balance in GeoGrid can be formalized as a multi-objective optimization problem. Suppose we are given a rectangular geographical plane $S$, which is defined as a region $<X,\ Y,\ W,\ H>$. We can define a continuous function $\ell(x, y),\ X \leq x \leq X + W,\ Y \leq y \leq Y + H$ to approximate the workload distribution of this region.

Given a region $r$ $<x_r,\ y_r,\ w_r,\ h_r>$, its workload can be calculated as:

$$\int_{x_r}^{x_r+w_r} \int_{y_r}^{y_r+h_r} \ell(x, y)\mathrm{d}x\mathrm{d}y$$

Suppose region $r$ is owned by a node $p$ with capacity $c_p$, the loading on this node can be measured with a metric *workload index*, which is defined as:

$$\frac{\int_{x_r}^{x_r+w_r} \int_{y_r}^{y_r+h_r} \ell(x, y)\mathrm{d}x\mathrm{d}y}{c_p}$$

Given a set of regions $R$ defined on $S$, a set of nodes $P$, and a node-region mapping function $A : R \rightarrow P$, we define the *mean of workload index* $MWI(S, R, P, A)$ as:

$$MWI(S, R, P, A) = \frac{\sum_{r \in R} \frac{\int_{x_r}^{x_r+w_r} \int_{y_r}^{y_r+h_r} \ell(x,y)\mathrm{d}x\mathrm{d}y}{c_p}}{\mid R \mid}$$

where $p = A(r)$;

and define the *standard deviation of workload index* $SWI(S, R, P, A)$ as :

$$SWI(S, R, P, A) = \sqrt{\frac{1}{\mid R \mid} \sum_{r \in R} (\frac{\int_{x_r}^{x_r+w_r} \int_{y_r}^{y_r+h_r} \ell(x, y)\mathrm{d}x\mathrm{d}y}{c_p} - MWI(S, R, P, A))^2}$$

where $p = A(r)$.

The problem of load balancing in GeoGrid can be formalized into a two-objective optimization problem:

$$min \quad MWI(S,R,P,A) \text{ and } SWI(S,R,P,A)$$

$$s.t. \quad MWI(S,R,P,A) = \frac{\sum_{r \in R} \frac{\int_{x_r}^{x_r+w_r} \int_{y_r}^{y_r+h_r} \ell(x,y)\mathrm{d}x\mathrm{d}y}{c_p}}{\mid R \mid}$$

$$SWI(S,R,P,A) = \sqrt{\frac{1}{\mid R \mid} \sum_{r \in R} (\frac{\int_{x_r}^{x_r+w_r} \int_{y_r}^{y_r+h_r} \ell(x,y)\mathrm{d}x\mathrm{d}y}{c_p} - MWI(S,R,P,A))^2}$$

This problem is hard to solve even when the global knowledge of node capacity distribution and workload distribution is given, because the standard deviation can hardly be mapped into a linear function. In a distributed environment like GeoGrid, global knowledge regarding node and workload distribution can hardly be obtained. To ensure the quality of service in GeoGrid, we propose a heuristic load balance scheme that can dynamically adjust the workload distribution. Such a scheme is composed of a number of techniques. *Dual Peer* technique improves the overall system reliability and generally maps the region sizes to the capacities of region owner nodes. *Load Adaptation* techniques include a number of adaptation mechanisms that can dynamically adjust the node-to-region assignment among regions in geographical vicinity.

To help us understand the design issues of GeoGrid, we develop a visualization tool. We use this tool to study the performance of different load balance mechanism of GeoGrid. To better illustrate the difference among different load balance mechanisms, we synthesize a simplified unbalanced workload distribution. In a geographical plane defined as $<X,\ Y,\ W,\ H>$, the workload distribution function is defined as $\ell(x,y) = \frac{x-X}{X}$. The capacity of the owner node of each region is printed in the upper left corner of it.

We use *Workload Index* to measure the workload of a region.

### 4.4.1 Dual Peer

We expect that the majority of nodes in a GeoGrid network are end systems owned by end-users. Like in the P2P networks used for file sharing, the continuity of service in a GeoGrid network is greatly affected by the dynamic member node population. Measurement studies like [50] reveal that peers in Gnutella networks have high turn-over rates. Half of them are replaced by new ones in an hour.

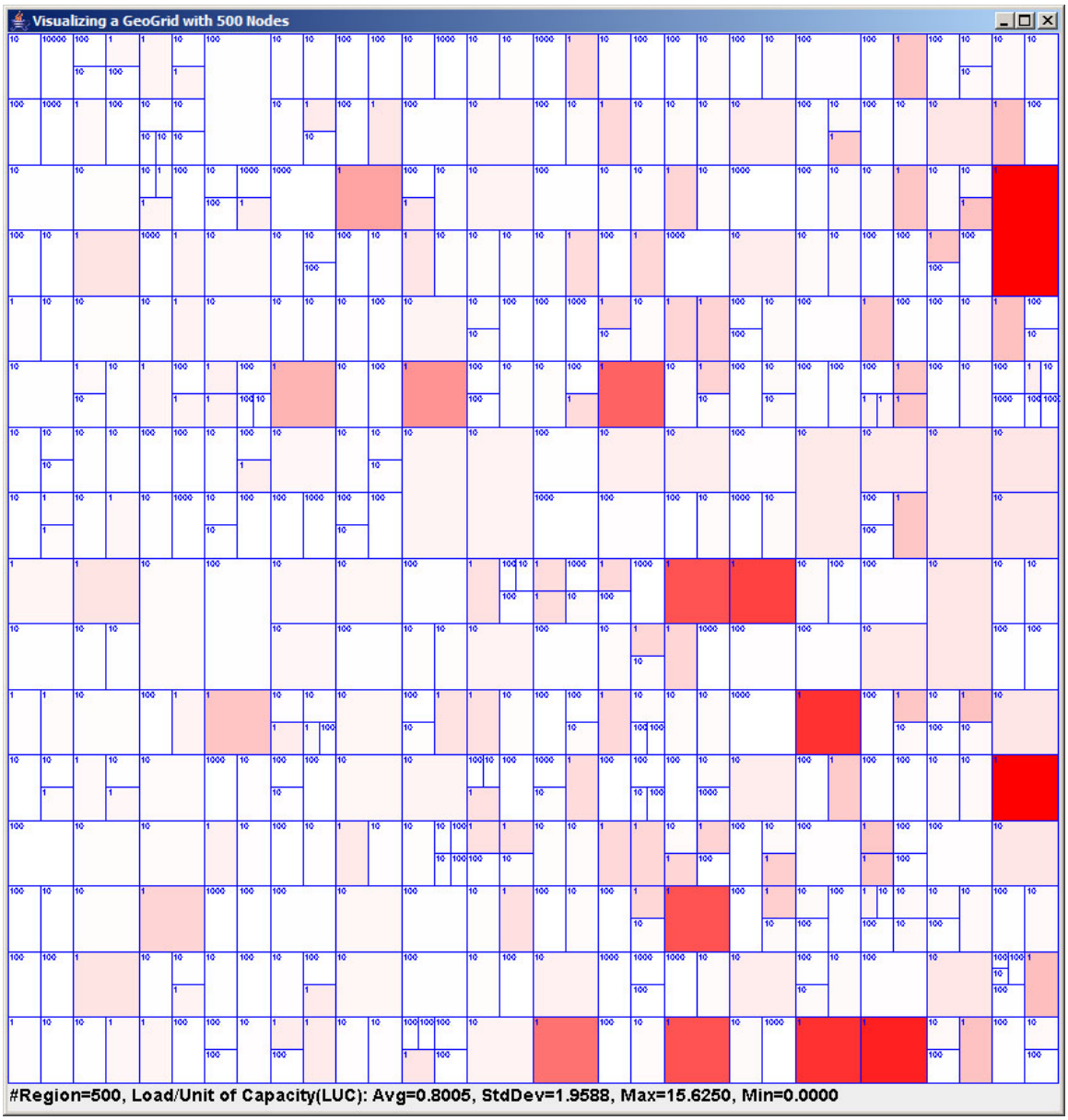


**Figure 52:** Region size and load distribution of GeoGrid, using random bootstrapping algorithm

For a service network like GeoGrid that relies on end system nodes for storage of objects and forwarding messages, such network dynamic could cause loss of information, service interruption, and extra communication and computation overhead for restoring the services. Passive replication techniques have been studied in the literature to accommodate such system dynamics [63, 29, 9, 32] . By replicating critical information on selectively chosen backup nodes, the loss of information and service interruptions can be considerably reduced.

Information and services can be directly recovered from those replication, in the case of node failure. Our experimental and analytical studies of Chapter 2 show that, by keeping even a single service replication, the fault resilience of the service network can be greatly improved. We incorporate a similar idea in GeoGrid based on the results of Chapter 2.

Instead of using a single node to manage the objects and to handle requests in a geographical region, we allow two nodes to share its ownership and to store the information published in it. The node with more capacity serves as the *primary owner* node, and handles all the requests and stores information in the way described in Section 4.2. The other node, what we call the *secondary owner* node, will serve as the backup of the primary node, holding the replication of objects and application-specific data copied from the primary node.

The basic GeoGrid needs to be modified to support the dual peer mechanism. Concretely, the following three aspects of the system design are modified.

**Node Join** The first three steps a new node $p$ follows to join a GeoGrid network featured with dual peer mechanisms will be the same as the basic GeoGrid, i.e., i) the new node $p$ obtains its geographical coordinate, ii) $p$ uses a bootstrapping service to randomly choose an existing node as its entry point, iii) $p$ uses the routing interface of GeoGrid to locate the region $r$ that covers its geographical coordinate. After the new node obtains the information of the primary owner $r.primary$ of region $r$, it will not directly split $r$ with the existing primary or secondary node. Instead, it will probe the neighbor regions $r.neighbors$ of $r$, and will choose from $r.neighbor \cup r$ a region that is not full and the owner of which has the least available capacity. If all the regions in $r.neighbor \cup r$ are full, $p$ will choose and split the region whose primary node has the least available capacity. Between the two new regions generated by the splitting, node $p$ will join the one whose owner has less available capacity. When node $p$ joins a region that is half full, it will compare its capacity with the capacity of the existing owner, and will take over the role as the primary owner if the current owner has less capacity than it. The switching of primary and secondary ownerships will happen after the new node finishes copying all the objects and status information from the existing

owner. The service interruption and loss of information caused by handover will have little impact on the continuity of the services.

**Node Departure** If a region is full, i.e., has two owner nodes, the departure of the secondary owner will cause no change to the GeoGrid system other than triggering the primary node to mark this region as "half full". The departure of the primary owner will cause the activation of the secondary owner. The new owner will inform the neighbor regions of the change and ask them to update the routing information of their primary and secondary owners.

**Failure Recover** The primary and secondary nodes of a region periodically synchronize their status information and exchange heartbeat messages at a higher frequency then among the primary nodes of different regions. The failure of a node will leave its region with one or no owner. If a region is full and its primary owner node fails, the secondary owner node will take over its role, activate all the backup information, and notify the neighbors and other nodes of such a change. If the failing node is the last owner of a region, the repairing process of the basic GeoGrid network will be triggered. Otherwise, the region will be left half-full and will be filled by a node joining later or the load balancing adaptation scheme that we will discuss later in this section.

The dual peer feature gives GeoGrid three advantages.

First, it improves the fault resilience of the GeoGrid service network. In our basic design, if a node fails, the objects held by it will have to be republished by the information provider nodes. For applications like end system multicast and publisher/subscriber systems, the application status information are usually stored on each node as soft-state information. The failure of a node is usually detected by the timeout of soft states pointed to this node. The lost application status information is usually restored by repeating the subscription or service announcement process. Dual peer technique can help avoid the extra processing time as well as the delay caused by the failure of nodes. When a secondary owner takes over the role of the failed primary owner, there is no need to reinsert or restore the application status and object information, and there will be less service interruption caused by the

failure of the primary node.

The second advantage of dual peer technique is that it reduces the number of region split operations that may causes service interruption in the basic GeoGrid systems. A new node will always try first to join a half full candidate region. No matter it takes over the role as the primary owner or as the secondary owner of a originally half full region, it will not force the split of a region and thus only need to copy the object information and application status information from the existing owner. When a new node has to split an existing region that is originally full, it will only need to copy half of the information from one of its existing owners. Before the new node finishes the copying, the original primary and secondary owner nodes can serve as the primary owners of the two new regions. After finishing the copying, the new node will then start taking the role as a primary owner or a secondary owner, depending on its capacity and the capacity of the current owner nodes.

The third advantage of dual peer technique is in improving the system load balance. A new node probes existing neighbors of the region that covers its coordinate, and joins or splits the region with the weakest primary node. Such a process will leave the regions owned by powerful nodes split fewer times and will reduce the size of the regions owned by weaker nodes. Figure 53 is the visualization of a GeoGrid service network of 500 nodes. Comparing it to Figure 52, we have two observations. First, there are fewer regions and the sizes of them are distributed in less uniform manner, conforming to the capacity distribution of owner nodes. More powerful nodes now own bigger regions. Second, the selective joining process of duel peer technique renders fewer heavily loaded regions, although a few still exist.

Dual peer technique can be further improved in two aspects.

First, we can use a pool of nodes to replace the single secondary node. The basic GeoGrid design and one featured with dual peer technique will add all the end system nodes in a geographical area into GeoGrid service network either as a primary owner or secondary owner. When a geographical region has a dense end-user population, each region may be too small to make the application-level communication efficient. For example, in a downtown area of one square mile, is likely that there are thousands of end-users. Each of

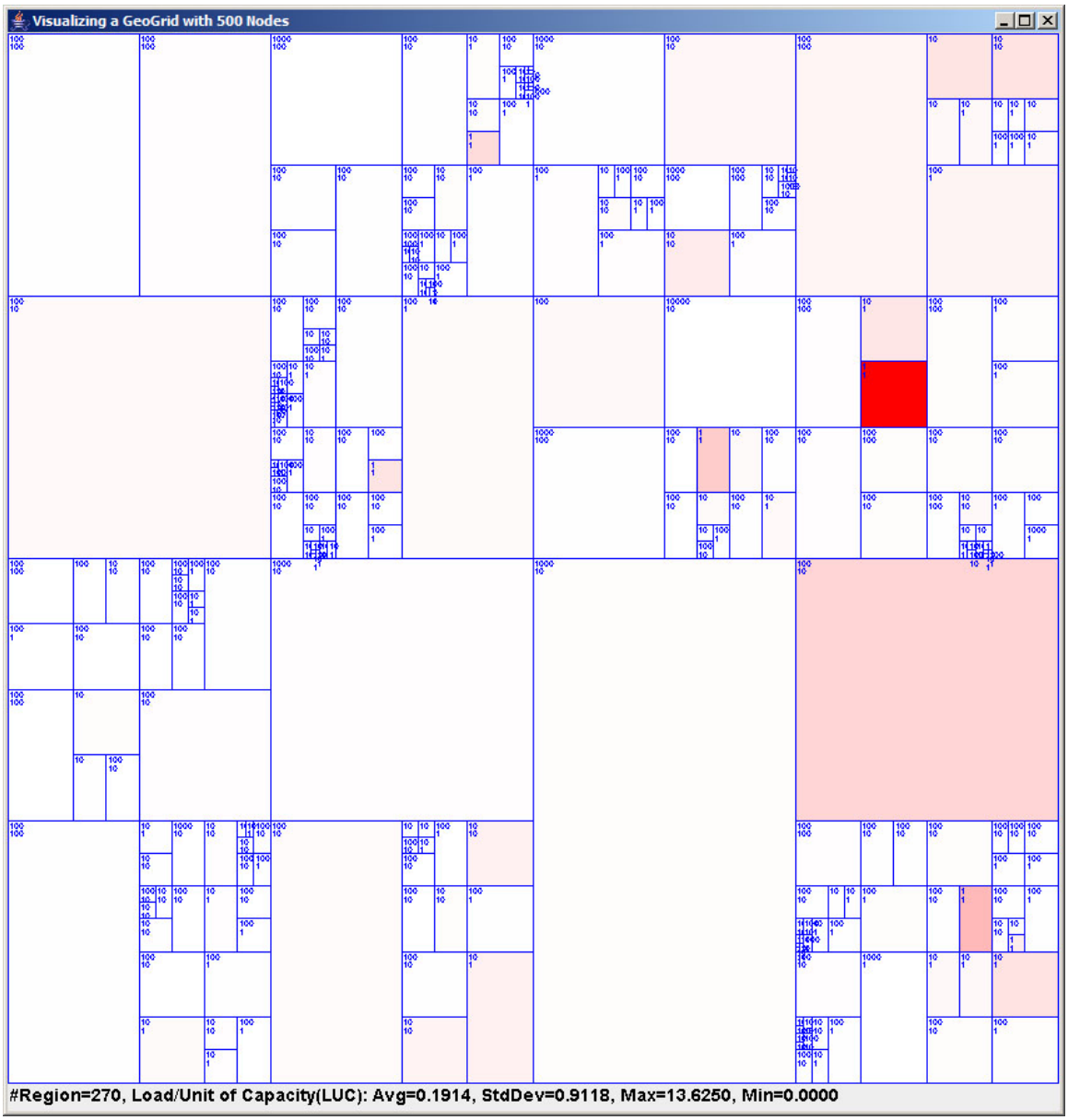


**Figure 53:** Region size and load distribution in the GeoGrid featured with the dual peer technique

their nodes will handle a region of only hundreds of square yards. The routing among those regions is more expensive because more routing hops are involved. Furthermore, processing of area-based queries will involve much more nodes than necessary. By posing a lower bound on the region size, we can reduce the total number of regions in a small geographical area. When a node tries to split a region that has reached this size lower bound, it will join the secondary pool of that region, if it is weaker than the current primary owner

node. Otherwise, it will take the role as the primary owner of the region and put the current primary owner into the secondary owner pool. The secondary pool will be used in a way similar to the original secondary peer. Besides taking the role of secondary peer, the secondary pool is more helpful on the load balancing of GeoGrid. When a GeoGrid service network is system-wide overloaded, the mechanism that we will present in Section 4.4.2 will be of limited help. In such case, we can reduce the value of region size lower bound, and promote more nodes in the secondary pool to join the GeoGrid service network. By trading the routing efficiency with more processing power, we will reduce the average workload handled by each active owner node in the GeoGrid service network.

The second technique we use is to randomize the geographical coordinate of a joining node in a high density area. Concretely, each node will add randomly generated delta ($\Delta x,\ \Delta y$) to its geographical coordinate ($x,\ y$) when joining a GeoGrid service overlay. The new coordinate ($x+\Delta x,\ y+\Delta y$) will determine the region that the new node will join or split. By doing so, we can spread nodes from a dense region more evenly, and improve the load balance in GeoGrid service network.

### 4.4.2 Dynamic Work Load Adaptation

The GeoGrid visualized in Figure 53 still has a few heavily loaded regions, even though their number is smaller than that of Figure 52. Those overloaded regions all have relatively weaker primary owner nodes. Dual peer technique can balance workload distribution by selectively assigning new nodes to the most heavily loaded regions in the neighborhood of the new nodes. However, when the nodes in a region are all weak ones, the effects of dual peer scheme will be less significant.

To further improve the system load balance of GeoGrid service networks, we develop a set of adaptation mechanisms. The basic idea behind those adaptation mechanisms is to break the geographical association between an owner peer and the region it owns, and dynamically adjust the node assignments in a geographical vicinity according to the workload distribution.

Figure 54 illustrates eight adaptation mechanisms we use in GeoGrid. Each of them

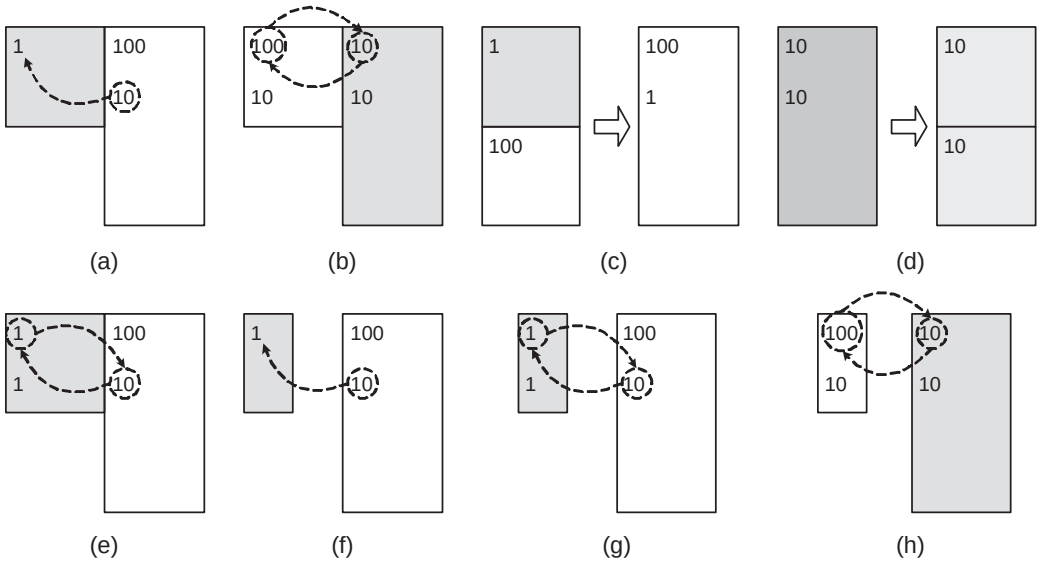


**Figure 54:** Load balance adaptation mechanisms of GeoGrid

describes an adaptation scenario. The capacities of the primary and secondary owners are printed in the upper left corner of each region.

Before we explain the load balance mechanisms, we first define the condition that triggers the load balance adaptation. In a GeoGrid network, each node periodically exchanges workload statistic information with its neighbors. A node starts its load balance adaptation process only when its workload index is higher than $\sqrt{2}$ times of the lowest one among its neighbors. By doing so, we can avoid the load balance adaptation process being repeatedly triggered within a geographical area in a certain time window.

Once the load balance adaptation condition is satisfied, one of the following eight mechanisms will be used to adjust the owner node assignments. Algorithm 2 provides the guideline that nodes in GeoGrid follow to conduct those adaptations. The basic rules are:

- local adaptations have less operation overhead than remote adaptations, and thus have higher priority.
- switching or moving secondary peers has less operation overhead than switching or moving primary peers.
- splitting and merging are expensive operations and are thus assigned with the lowest priority.

**Steal Secondary Owner** This adaptation is used when the overloaded region is not full. Using this adaptation, the overloaded primary owner node compares the workload index of all the neighbor regions. It tries to select a neighbor region whose secondary owner is more powerful than it, and has the lowest workload index among all the regions satisfying the first condition. Once such a region is located, its secondary owner is "stolen" to be the primary owner of the overloaded region. An example is given in Figure 54(a). The region in gray shade has a primary owner with capacity 1. It steals the secondary owner of a neighbor region, which has capacity 10, to replace its position as the primary owner. After the adaptation, the primary node of the originally overloaded region now has more capacity to handle the workload assigned to it.

**Switch Primary Owners** This adaptation can be initiated by a region that is either half-full or full. Figure 54(b) gives an example. A smaller region has a primary owner that is more powerful than one of its neighbor regions, which is bigger and has a weaker primary owner. By switching the primary owners of these two regions, the bigger region now has more processing power while the smaller one has less. The assignment of processing capacities now matches the sizes of both regions better.

**Merge with a Neighbor** This adaptation is used when a region $p$ and one of its neighbor region $n$ can be merged, and the merged region has lower workload index than the average workload index of $p$ and $n$. An example is given by Figure 54(c).

**Split a Region** As illustrated in Figure 54(d), if the primary and secondary owner of an overloaded region have the same capacity, splitting this region can assign half of the workload to each of them and can reduce the workload index of the original primary owner by half. The two half-full regions are left to be filled by the other adaptation mechanisms or be filled by nodes that join the network later.

**Switch Primary and Secondary Owners** When an overloaded region is full, its primary owner can switch its position with a secondary owner of a neighbor region, if that

**Algorithm 2** Load balance adaptation

```
1: procedure LoadBalanceAdaptation *defined for each node p*
2:     if needAdapation()==true then
3:         if isNotFull() then
4:             if (n=searchSecondary())!= null then
5:                 stealSecondaryFrom(n)
6:                 return
7:             end if
8:             if (n=searchPrimaryToSwitch())!= null then
9:                 switchPrimary(n)
10:                return
11:            end if
12:            if (n=searchRemoteSecondary())!= null then
13:                stealSecondaryFrom(n)
14:                return
15:            end if
16:            if (n=searchRemotePrimary())!= null then
17:                switchPrimary(n)
18:                return
19:            end if
20:            if (n=searchMergibleNeighbor())!= null then
21:                merge(n)
22:                return
23:            end if
24:        else *this region is full*
25:            if (n=searchSecondary())!= null then
26:                switchPrimarySecondary(n)
27:                return
28:            end if
29:            if (n=searchPrimaryToSwitch())!= null then
30:                switchPrimary(n)
31:                return
32:            end if
33:            if (n=searchRemoteSecondary())!= null then
34:                switchPrimarySecondary(n)
35:                return
36:            end if
37:            if (n=searchRemotePrimary())!= null then
38:                switchPrimary(n)
39:                return
40:            end if
41:            if primary.capacity() == secondary.capacity() then
42:                n = split()
43:                n.join()
44:                return
45:            end if
46:        end if
47:    end if
48: end procedure
```

secondary owner has more capacity. An example of such adaptation is given by Figure 54(e).

**Steal Remote Secondary Owner** There is non-zero probability that a region and all its neighbors are all overloaded. Local adaptations through switching or stealing owner nodes from neighbor regions may not improve the distribution of workload. In such a case, we consider using nodes that are not direct neighbors of the overloaded region for adaptation. However, a new problem stands out. How can an overloaded node discover a remote node for adaptation? In our system design, we have considered two approaches. The first one is by flooding query messages to the neighbors, like in a Gnutella P2P network. The drawback of this approach is that a large number of redundant messages will be generated. Some of them are sent to nodes that are already overloaded and will further increase their workloads. The second approach, which is the one we adopt in GeoGrid, is based on a guided search. At each step of the search, the query message is forwarded from a node to its least loaded neighbor. The search terminates when the stop condition is satisfied or the Time to Live (TTL) of the query runs out. The initial value of TTL is set to 5. If a search fails, the TTL of the next search will be increased by 1. A success search resets the value of the initial TTL of next search to 5.

The adaptation illustrated by Figure 54(f) is based on such a search. The stop condition is that the search reaches a region whose secondary owner has more capacity than the primary owner of the overloaded region and is less loaded. Assume a remote secondary owner is discovered. The primary owner of the overloaded region will steal this remote secondary owner, and will resign to be the secondary owner.

**Switch Primary with Remote Secondary Owner** This adaptation is for a full region and is also based on a search for discovering a remote secondary owner that is stronger than the primary owner of the overloaded region. The overloaded primary owner will switch its position with the discovered remote secondary owner, as shown in Figure 54(g).

**Switch Primary with Remote Primary Owner** This adaptation is for a full region and is also based on a search for discovering a candidate remote primary owner that is

stronger than the primary owner of the overloaded region. The overloaded primary owner will switch its position with the discovered remote primary owner, as shown in Figure 54(h).

Please note that each time a node launches its adaptation process, it will begin with the least expensive operation. Expensive ones like switching primary owner with remote primary owner are used only when all the other adaptations fail.

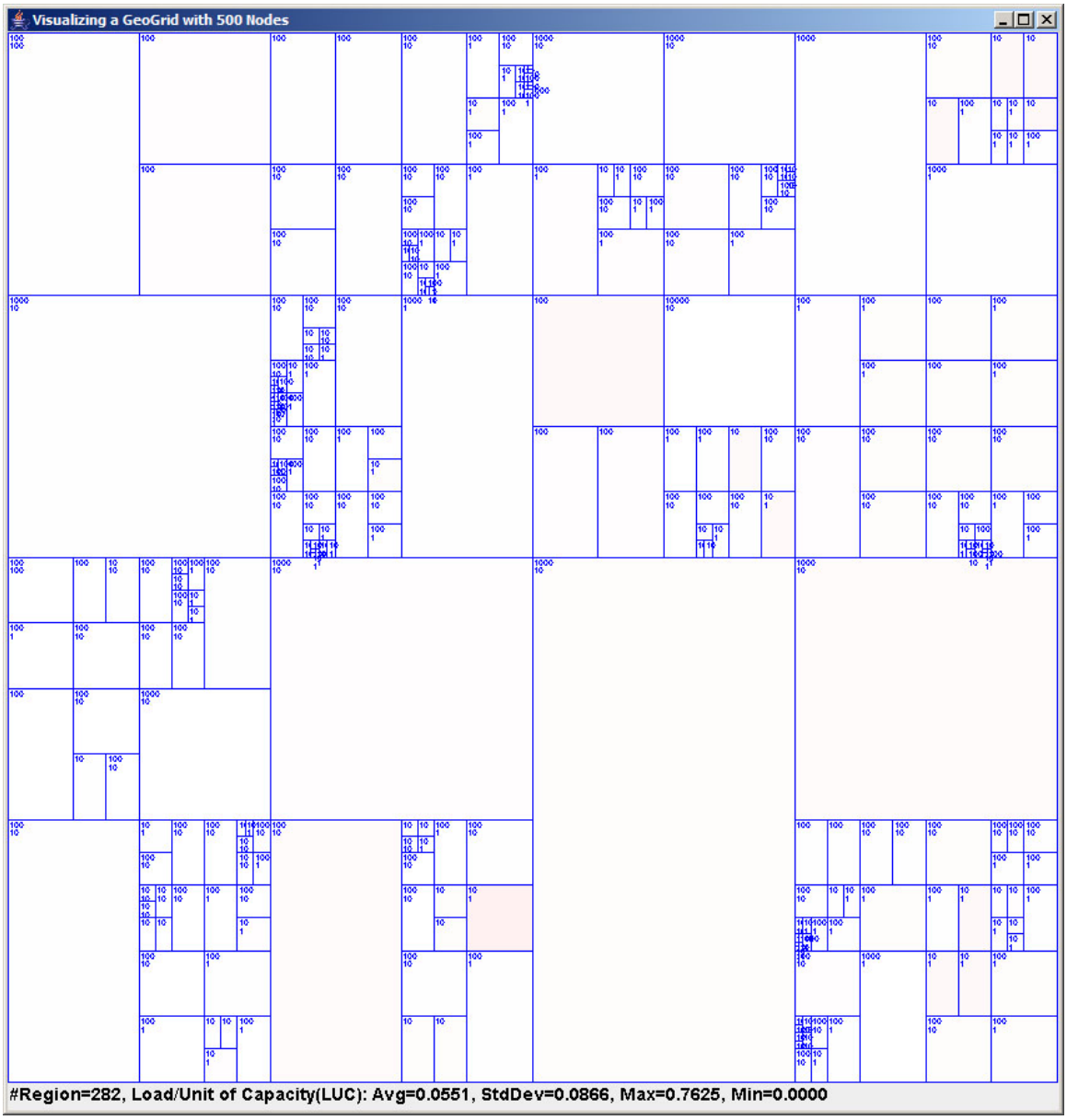


**Figure 55:** Region size and load distribution in the GeoGrid featured with both dual peer and load balancing adaptation techniques

Figure 55 visualizes a GeoGrid network adapted from the one presented by Figure 53.

We can see that all the overloaded nodes are replaced with stronger ones, and the max, average, and standard deviation of workload index are all significantly lower than the ones of GeoGrid featured with only dual peer technique.

## *4.5 Location-Based Routing in GeoGrid*

In a GeoGrid network of $N$ regions, the average number of routing hops between two randomly selected regions is $O(2\sqrt{N})$. For a GeoGrid network with a large number of nodes, such routing performance is less desirable than the $O(\log N)$ ones of Chord [52] and Pastry [49]. The less efficient routing performance brings two performance issues to our attention. The first one is the long delay of communication. Secondly, the more hops a message is forwarded through, more nodes in the GeoGrid service network will be involved in processing and forwarding the message. Consequently, more routing workload will be placed onto the GeoGrid network and less capacity will be available for processing application-related messages and objects.

Furthermore, when choosing a neighbor to forwarding a routing request, the only criteria used in basic GeoGrid design is the distance to the destination point. Although the dual peer and dynamic load balance adaptation techniques we presented in Section 4.4 can help balance the workload distribution in GeoGrid, the unbalanced routing workload distribution is still an outstanding issue.

To solve this problem, we propose a randomized shortcut technique that can improve both the routing efficiency and routing workload balance in GeoGrid.

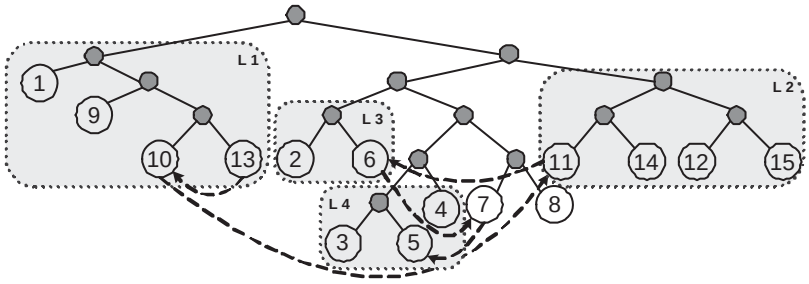


**Figure 56:** Binary tree representation of GeoGrid and routing

### 4.5.1 Accelerate Routing with Shortcuts

In the basic GeoGrid design, a routing message is always forwarded from one region to one of its neighbors that is the closest to the destination point. Although this mechanism is deterministic and intuitional correct, some hops in the routing path are redundant. We can think of each region in a GeoGrid service network as a leaf of a binary "partition tree". The internal vertices represent regions that have been split. The children of a tree vertex are the two regions into which it was split. Although this partition tree is not maintained in GeoGrid system and is totally conceptual, it can help us analyze the routing performance of GeoGrid service networks. Figure 56 presents the partition tree of the GeoGrid network given by Figure 50. The dashed arrow lines represent the message routing path from region 13 to region 5. We find that although region 10 is the closest to region 5 among all the neighbors of region 13, using this hop to send a message into region 13 does not help reducing the search space, because region 10 and region 13 are siblings in the same subtree of the partition tree.

#### *4.5.1.1 Structure of Shortcut List*

One of the key techniques for reducing routing hops is by trading status information for efficiency. Chord achieves $O(\log_2 N)$ routing efficiency because forwarding a routing message according to the finger table entries of each node can reduce the search space by half at each forwarding step. In Pastry, the search space is reduced to $\frac{1}{2^d}$ of the previous because each node of Pastry maintains even more routing information than Chord.

Inspired by this observation, we design a technique call *Routing Shortcut*. As implied by its name, the main idea is to add more routing information to each node, such that these routing entries can be used as the shortcuts in forwarding routing requests.

The shortcuts of a node $p$ is organized in to a list $L_p < s_1,\ s_2,\ s_3,\ \ldots >$. Shortcut $s_i$ points to a node in a geographical partition of $1/2^i$ the size of the geographical plane. There are no overlapping among the partitions pointed to by the shortcuts of $p$. When shortcuts of each GeoGrid node are setup in such a structured manner, each time when a node forwards a routing request, the shortcut it uses can effectively reduce the routing

search space by half.

The length of the shortcut list $L_p$ of a node $p$ is decided by the relative size of region $r$ owned by $p$. When region $r$ is $1/2^m$ of the size of the geographical plane, the length of the shortcut list $L_p$ is $m$. By doing so, the whole geographical plane is covered by the shortcuts of $p$, since we have the following equation:

$$\sum_{i=1}^{m} \frac{1}{2^i} + \frac{1}{2^m} = 1$$

Based on this analysis, we can estimate the average length of the shortcut list maintained by each node. In a GeoGrid service network of $N$ nodes, the size of a region is $\frac{1}{N}$ of the geographical plane, assuming a uniform region size distribution. Thus the length of the shortcut list maintained by each node is $O(\log_2 N)$.

One challenging issue we need to address is to choose the proper geographical partitions covered by each shortcut of each node. In GeoGrid, we exploit the structure of the conceptual binary partition tree for setting up shortcut list. We first define the *partitioning subtrees* of a region $r$ owned by node $p$. Concretely, the partitioning subtrees of a region $r$ are a list of subtrees of the binary partition tree that contain region $r$. We denote them as $< R_1, R_2, \ldots >$, where $R_i$ is the subtree at depth $i$ in the conceptual binary partition tree of the geographical region. We define the *shortcut subtrees* as a list of subtrees of the binary partition tree, and denote them by $< L_1, L_2, \ldots >$. $L_i$ is a subtree of the binary partition tree and is the sibling of $R_i$ at depth $i$.

Use region 7 in Figure 56 as an example. It maintains shortcut links pointing to nodes in shortcut subtrees L1, L2, L3 and L4. Each of those subtrees covers a geographical region that is the half the size of the region at the higher level. Figure 57 illustrates the geographical regions covers by those subtrees. For each subtree, one shortcut is maintained on the owner node of region 7.

Using these shortcuts, owner node of region 7 can route messages to the other regions with less hops. For example, when it routes a message destined to region 10, it can use one of its shortcuts to first send the request into regions covered by subtree L1, and effectively

reduces the search space by half. The node that receives this request will also use its shortcuts and neighbor list to further forward the request. In the whole routing process, the search space is reduced by half at each routing step.

#### *4.5.1.2 Selection of Shortcuts*

The standing out issue is how a node can obtain the information of shortcut entries in a distributed manner. It can be done in two different ways.

In the first approach, a node starts setting up shortcuts until it fully joins the GeoGrid service network. The setup can be done by sending out a number of routing requests. Each request is tagged with a random coordinate covered by one of its shortcut subtrees. The request is routed using the basic routing scheme of GeoGrid. The nodes that own the regions covering those coordinates send their information back to the new node. The information of those nodes will be stored in the shortcut list of the new node. Although this approach is straightforward, it requires extra probing messages to setup the shortcuts.

In GeoGrid, we adopt the second approach, which does not require such a probing process. Concretely, a new node uses the same process as in Section 4.2 to join a GeoGrid overlay. If it assumes the role of the primary owner of a region, it must have split an existing region and copied all the routing information from the owner of that region. Such routing information includes the shortcut information and the neighbor list. In our basic GeoGrid design, when a region is split, its owner node will discard the routing information of those regions who are no longer its direct neighbors. In our design with shortcuts, we add such routing information into the shortcut list, instead of discarding it. The split history of a region records its path sinking from the top of the binary partition tree to the bottom leaf. Those routing information recorded by the shortcut list are the ones inherited from the existing nodes, and record the neighbors of partition subtrees at different depth. They are exactly the shortcuts that we are looking for.

#### *4.5.1.3 Utilization and Maintenance of Shortcuts*

The routing process using shortcuts is similar to the one we discussed in Section 4.2. The only change we make is to replace the neighbor list with the super set of neighbor list and

shortcuts. When a node evaluates the candidate nodes for forwarding a routing request, it will consider the entries in both neighbor list and shortcut list, and will choose the one closest to the destination point to forward the request.

Maintenance of shortcuts is similar to the maintenance of neighbor list. Heart beat messages are used to detect failures, though in a lower frequency. When a node detects a failed shortcut, it will initiate a routing request tagged with the coordinate of the shortcut node. The message will be forwarded to the node who takes over the region left by the failed shortcut node. And that new owner node will be used to replace the failed shortcut.

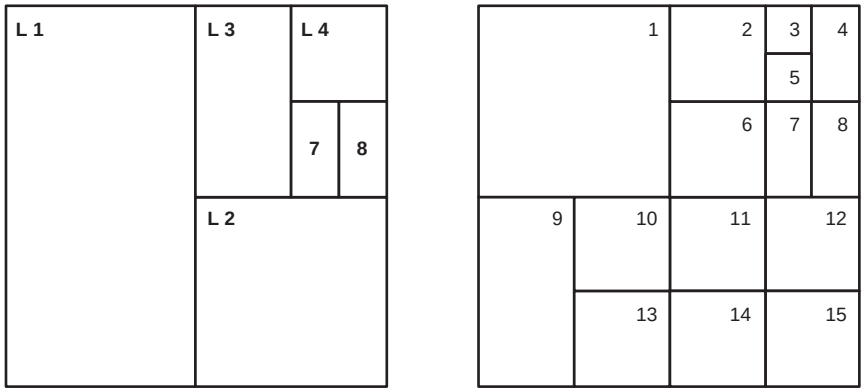


**Figure 57:** Geographical regions corresponding to the shortcuts

### 4.5.2 Randomizing Adaptation of Shortcuts

The way we setup shortcuts in GeoGrid can reduce the messaging overhead for setting up shortcuts. However, there is a potential problem with this approach. In the split history of a node, the shortcuts pointing to the shortcut subtrees of lower depth value are the ones recorded originally when the total population of the GeoGrid service network is small. As more nodes join and later depart the GeoGrid service network, such setup may cause a severe performance problem. The shortcuts of a node are inherited from the owner with whom it splits a region, and later are passed on to other nodes that split its region. As more nodes join the network, a lot of them may share the same shortcut links pointing to those nodes that have been in the GeoGrid service network for a while. When those shortcuts are used to accelerate routings in a GeoGrid service network, nodes with longer lifetime may be used more by other nodes for routing. If those "senior" nodes do not have enough capacity

and is overloaded, the shortcuts pointed to them will be of no use, and a lot of nodes in the GeoGrid service network may have to resort to the basic routing scheme for message forwarding.

To address this issue, we propose a technique to randomize the shortcuts such that nodes with more capacities are more likely be used as shortcuts and the distribution of shortcuts are also randomized. Concretely, our solution includes two parts.

First, we propose a shortcut selection algorithm. Each node running GeoGrid middleware will periodically use this algorithm to choose a shortcut from its shortcut list for adaptation. For a node $p$ with shortcut list $L_p$, $p$ calculates its geographical distance $d(p, s_i)$ to each shortcut entry $s_i \in L_p$. A shortcut $s_k$ is chosen with probability $\frac{d(p,s_k)}{\sum_{s_i \in L_p} d(p,s_i)}$. Node $p$ will make a random decision on whether a chosen shortcut neighbor node $s$ should be replaced. The capacity of $s$ is compared to the product of the capacity of $p$ and a uniform random number Unif(0, 1). Concretely, if $s.capacity \leq$ Unif(0,1) $\times$ $p.capacity$, $s$ will be replaced. Such a design tends to keep the shortcut neighbors that have more capacity, and replace the weaker ones with higher probability.

The second component of our solution is an algorithm for replacing the selected shortcut. We want to put the node capacity into consideration when we replace shortcuts. Intuitively, a node with more capacity should handle more routing workload. Concretely, each selected shortcut is replaced by a randomly selected node in the same shortcut subtree. The replacement shortcut is chosen by a selective random walk. The random walk starts from the node chosen to be replaced. If this node is no longer in the GeoGrid network, its coordinate will be used to locate the new owner node who takes over its region. The random walk is tagged with the information of the node $p$ that initiates the shortcut replacement algorithm. At each step of the random walk, a node $n$ receives the random walk query message from a node $n'$. It will first generate a random number Unif(0,1) and compare $n.capacity$ to Unif$(0, 1) \times n'.capacity$. The random walk stops when $n$ is a local optimal, i.e., it has more capacity than $n'$. If the stop condition is not satisfied, node $n$ will select one of its direct neighbors for the next hop of random walk query. A neighbor node $n_k$ is chosen with probability $\frac{n_k.capacity}{\sum_{n_i \in n.neighbors} n_i.capacity}$. By doing so, the random walk is directed to and

**Table 2:** Capacity distribution of peers

| Capacity level | Percentage of peers |
|---|---|
| 1x | 20% |
| 10x | 45% |
| 100x | 30% |
| 1000x | 4.9% |
| 10000x | 0.1% |

terminates on the nodes with more capacity.

## *4.6 Experimental Results*

We evaluate GeoGrid system by simulating a geographical region of 64 miles $\times$ 64 miles. The population of end users in this region ranges from $1 \times 10^3$ to $1.6 \times 10^4$. For each population setting, we simulated 100 randomly generated GeoGrid service network. Each end user connects into the GeoGrid service network through a dedicated proxy node. The capacities of those proxies follow a skewed distribution borrowed from a measurement study of Gnutella P2P network [50].

### 4.6.1 Routing Efficiency

We first study the routing performance of GeoGrid system. As we discussed in Section 4.5, the average number of routing hops in a basic GeoGrid system with $N$ nodes is of $O(2\sqrt{N})$. We add shortcuts to each nodes and expect to lower this number to $O(\log N)$ and also to reduce the average routing workload on each node.

We simulate the routing in three types of GeoGrid systems, i.e., the basic GeoGrid system, the GeoGrid system with shortcuts added, and the GeoGrid system with shortcuts that have been randomly adapted. For a GeoGrid system with $N$ nodes, we simulate the processing of $2N$ routing requests. Each routing request is between a pair of randomly chosen nodes. We record the number of hops each routing request is forwarded through, and the number of forwarding handled by each node.

Figure 58 plots the average hops a routing request is forwarded through. We have two observations. First, shortcuts can effectively reduce the average number of routing hops from

$O(2\sqrt{N})$ to $O(\log N)$. This improvement makes the GeoGrid with shortcuts more scalable in handling routing requests. Secondly, the side effects of random adaptation on shortcuts are negligible. The bars in Figure 58 plot the magnified average hops of routing in two GeoGrid systems with shortcuts. The extra routing delay caused by shortcut randomization is between 2.18% and 4.97%.

Figure 59 plots the standard deviation of hops a routing request is forwarded through. It demonstrates similar patterns as Figure 58. Such similarity indicates that GeoGrid systems with shortcuts have more stable routing performance. And the side effects of randomizing shortcuts is more noticeable but still within an acceptable range.

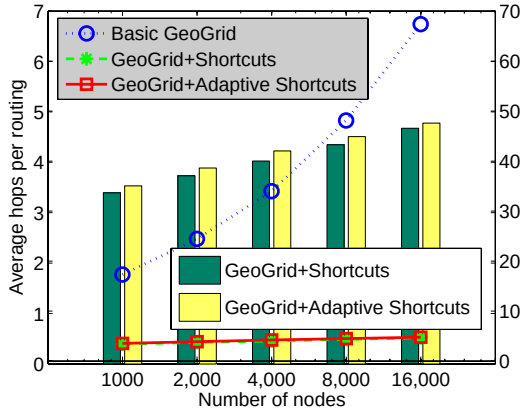


**Figure 58:** GeoGrid with shortcuts achieves $O(\log N)$ hops per request routing

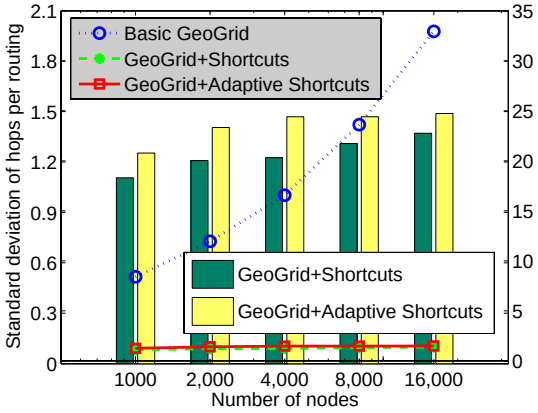


**Figure 59:** GeoGrid with shortcuts and adaptation has more stable routing performance

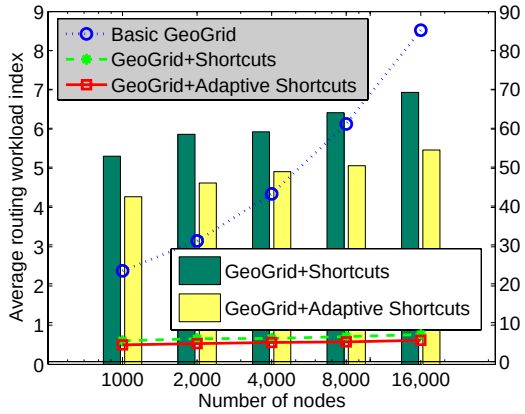


**Figure 60:** GeoGrid with shortcuts generates less routing workloads. Randomizing the shortcuts helps further reduce the routing workload index

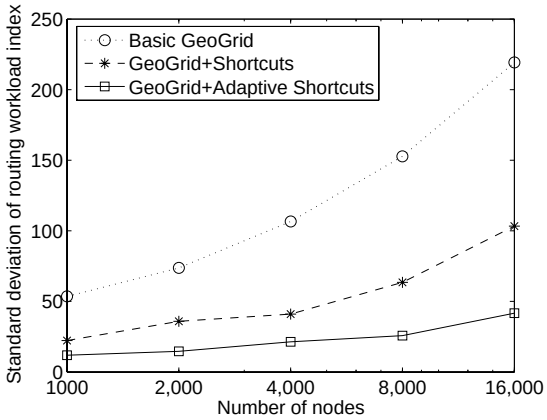


**Figure 61:** GeoGrid with shortcuts and adaptation distributes the routing workload more evenly among heterogeneous nodes

### 4.6.2 Balance Routing Workload

To measure the routing workload distribution, we record the accumulative number of routing requests handled by each node. We use a metric *Routing Workload Index* to measure the workload distribution. For a node $p$ with capacity $c$, its routing workload index is defined as the accumulative routing requests forwarded by each unit capacity of $p$.

Figure 60 plots the average routing workload index of different GeoGrid systems. We observe similar patterns as in Figure 58 and Figure 59. Shortcuts can effectively reduce the routing workload in the system because each routing request is forwarded through fewer hops and consequently requires less resource for handling. An interesting observation is

that the randomizing of shortcuts helps to utilize the heterogeneous node capacity more efficiently. The adaptation random walks make more shortcut entries pointing to more capable nodes and thus make them handle more routing requests in average. The result is that routing workload index of GeoGrid with adaptive shortcuts is around 17.13% to 21.18% lower than the GeoGrid with non-adaptive shortcuts.

While the average routing workload index is related with the total number of routing requests initiated in the GeoGrid overlay, we believe that the standard deviation of routing workload index is a better metric for measuring the routing workload distribution. Figure 61 plots the standard deviation of routing workload index of different GeoGrid systems. The effects of randomizing shortcuts are significant. This mechanism helps to reduce the standard deviation of routing workload index by 46.51% to 59.66%. The interpretation of such reduction is that routing workloads are more evenly distribution through randomizing the shortcuts.

### 4.6.3 Effects of Adaptation on Balancing Routing Workload

To further understand the effects of randomizing adaptation of shortcuts on both routing efficiency and routing workload distribution, we design a simulation on a randomly generated GeoGrid system with $8 \times 10^3$ nodes. This network is featured with shortcuts, but we disable the randomizing adaptation until the network is stable. Before we turn on the randomizing adaptation of shortcuts, we measure the routing performance and routing workload distribution by simulating $1.6 \times 10^4$ random routings. We simulated 40 rounds of adaptations. In each routing of adaptation, each peer randomizes one of its shortcuts. After each round of adaptation, we repeat the same set of routing experiments and record the routing performance and routing workload distribution. Figure 62 presents the changes on standard deviation of routing workload index. Figure 63 plots the series of change on average routing workload index. Figure 64 records the changes of average routing hops per request. Figure 65 records the changes of the standard deviation of routing hops per request.

The results of this experiment confirm our observation in Section 4.6.1 and Section 4.6.2.

The randomizing adaptation of shortcuts can help distribute the routing workload more evenly while cause negligible side effects on the routing efficiency.

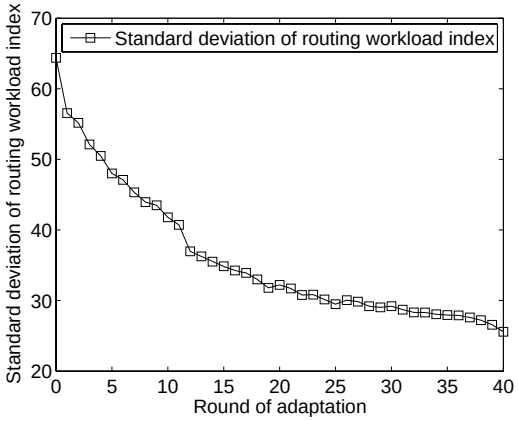


**Figure 62:** Reduce standard deviation of routing workload index using adaptation

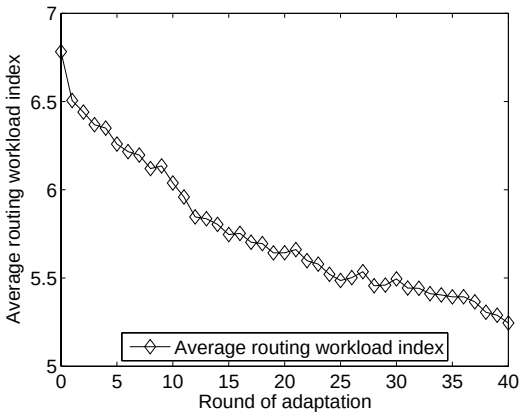


**Figure 63:** Reduce the average routing workload index using adaptation

### 4.6.4 Routing Performance under Clustered Geographical Node Distribution

In real world applications, it is unrealistic to expect nodes in GeoGrid to be uniformly distributed. We want to know if the routing performance of GeoGrid will degrade under uneven node distribution. We repeat the set of routing experiments on geographical planes with "skewed" end-node distribution. Figure 66 visualizes the node distribution we use. Nodes is a geographical plane are clustered in both latitude and longitude directions. Figure 67, 68, 69, and 70 plot the experimental results.

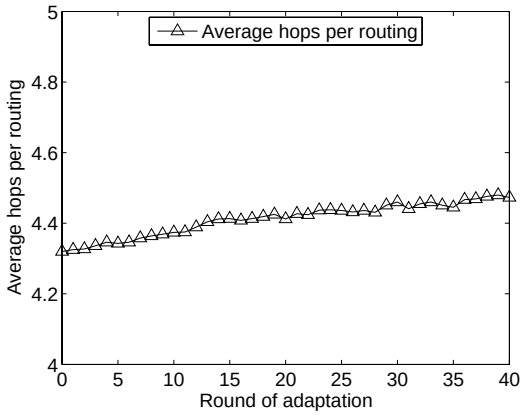


**Figure 64:** Minor impact of adaptation on the average number of routing hops

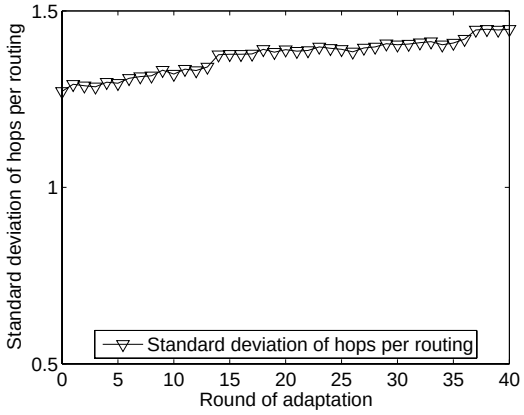


**Figure 65:** Minor impact of adaptation on the standard deviation of routing hops

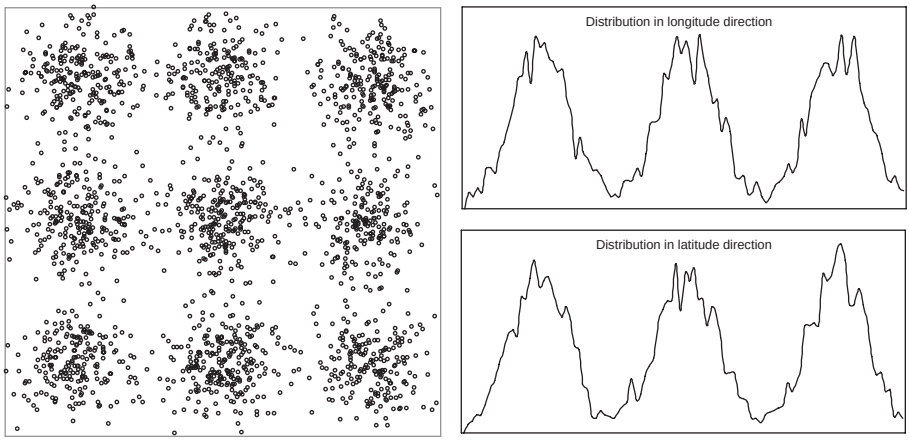


**Figure 66:** Skewed node distribution for evaluating routing performance of GeoGrid

We can see that GeoGrid system with shortcuts can still deliver stable routing performances. Adaptation of shortcuts moderately increases the average and standard deviation

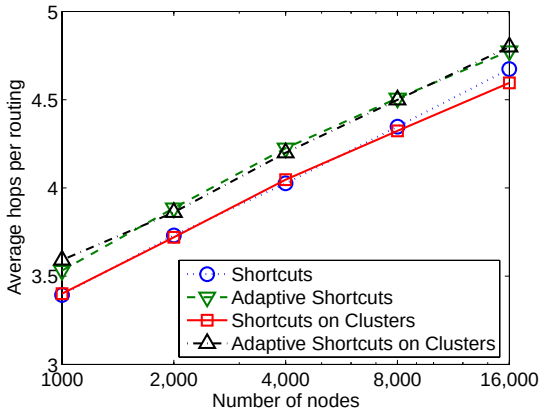


**Figure 67:** Minor impact of node distribution on the average number of routing hops

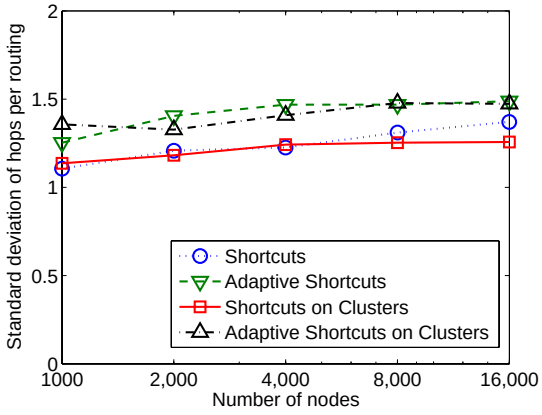


**Figure 68:** Minor impact of node distribution on the standard deviation of routing hops

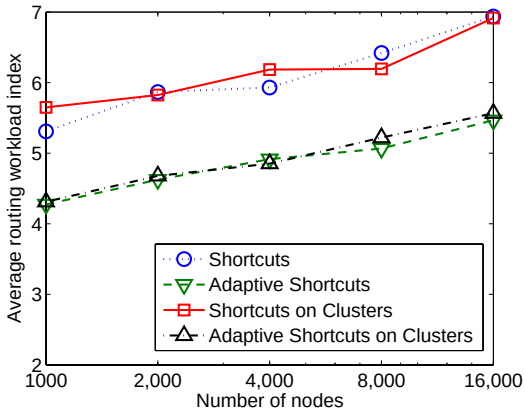


**Figure 69:** Minor impact of node distribution on the average routing workload index

of routing delay, but tremendously improves the routing load balance.

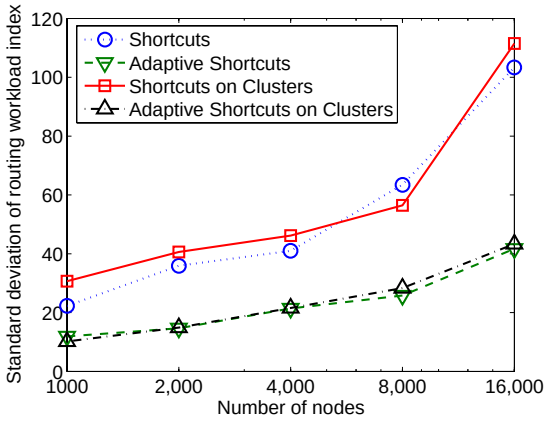


**Figure 70:** Minor impact of node distribution on the standard deviation of routing workload index

### 4.6.5 Balance Region Workload

We design a set of experiments to evaluate the effects of dual peer and load balance adaptation techniques we proposed in Section 4.4. We simulate the uneven workload distribution in the geographical area by creating a number of hot spots and randomly moving them in the simulated plane. Each hot spot is a circular area with a random initial radius between 0.1 and 10 miles. The cell at the center of a hot spot has the highest normalized workload 1 and the ones on its border have workload 0. The workloads of cells covered by the hot spot is decided by a formula $1 - d/r$, where $d$ is the distance of a cell to the center of the hot spot and $r$ is the radius of the hot spot. We choose circular hot spots because this choice agrees with the nature of location-based applications. To illustrate this point, let us use the queries on parking lots information as an example. Usually during a sport event like Super bowl, parking lots close to the stadium are usually fully loaded. More people will be interested in finding a parking space that is closer to the stadium. As we move from the stadium to the parking lots in the neighborhood, fewer people will be interested in querying them because parking there means longer walking to the stadium. Consequently, as the sport event creates a hot spot of queries in that area, more queries will be forwarded towards the center of the hot spot, and fewer will be forwarded to the nearby regions.

The whole simulation time line is divided into a number of eras. At the end of each era, we force each hot spot to migrate along a randomly chosen direction and at a random

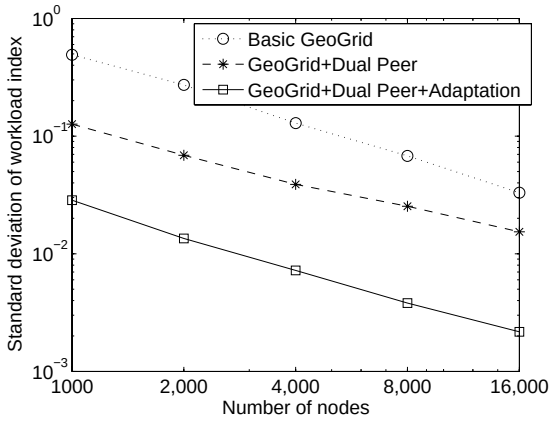


**Figure 71:** Standard deviation of workload index

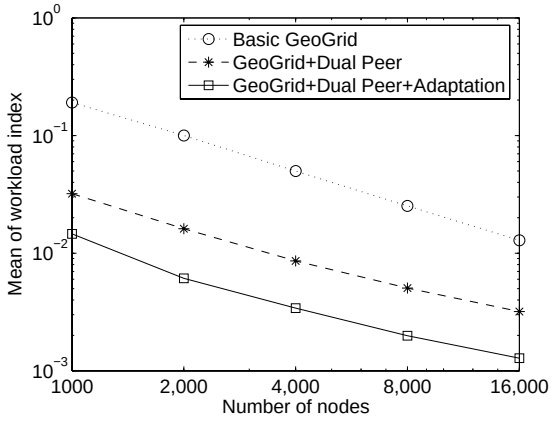


**Figure 72:** Mean of workload index

step size uniformly chosen from range $(0, 2r)$. We simulated three types of GeoGrid system: basic GeoGrid system, the one featured with dual peer technique, and the one with both dual peer and load balance adaptation turned on. We measure the max, mean, and standard deviation of workload index of each GeoGrid flavor. Each simulation is repeated 100 times on different node population.

As we observe in Figure 71 and Figure 72, both dual peer and adaptation techniques can effectively improve the load balancing in GeoGrid systems. The GeoGrid system with both features can constantly beat the basic GeoGrid system by one order of magnitude in both metrics. While dual peer technique itself can improve the workload distribution, the load balance adaptation can further improve the system performance.

The max of workload index demonstrate a similar patter as Figure 71 and Figure 72.

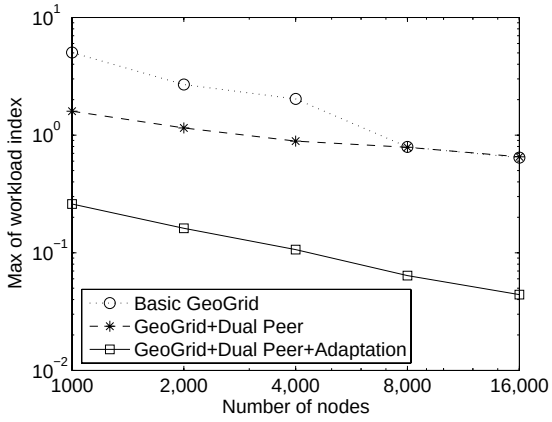


**Figure 73:** The max of workload index of the GeoGrid systems

We notice that when the node population is large, basic GeoGrid and the GeoGrid system featured only dual peer technique has similar max workload index. It is because we forces an upper bound $u$ on the number of splits allowed for a region in both GeoGrid systems. A region split operation fails when the region to be split has already been split $u$ times. The rollback of failed split will make the new node to split a randomly chosen neighbor region. When the total number of nodes is large, such a mechanism will generate similar system topologies for basic GeoGrid systems and the ones with dual peers. Thus, the max workload index tends to share similar values when the GeoGrid service network has a large number of nodes.

### 4.6.6 Impact of Adaptation on Region Workload

Both dual peer and load balance adaptation techniques can improve the workload distribution of GeoGrid systems. We want to know how fast load balance adaptation can improve the workload distribution. Will the adaptation converge? To answer this question, we design a simulation to evaluate the effects of load balance adaptation on GeoGrid systems. We simulate GeoGrid systems with $2 \times 10^3$ peers. The service network is setup first using only dual peer technique. When hot spots appear, we turn on the load balance adaptation features of each node. The max, mean, and standard deviation of workload index of all the nodes are recorded at the end of each round of adaptation. We simulated a "sunny" adaptation scenario in which the hot spots are static and never change their size or location.

The "rainy" scenario is simulated by constantly moving the hot spots at a pace far faster than the pace of adaptation. Concretely, hot spots move 4 to 10 steps before a round of adaptation ends.

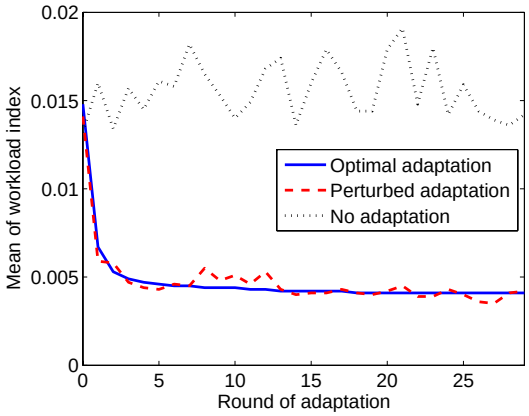


**Figure 74:** Convergence of the mean workload index in adaptation, plotted by round of adaptation

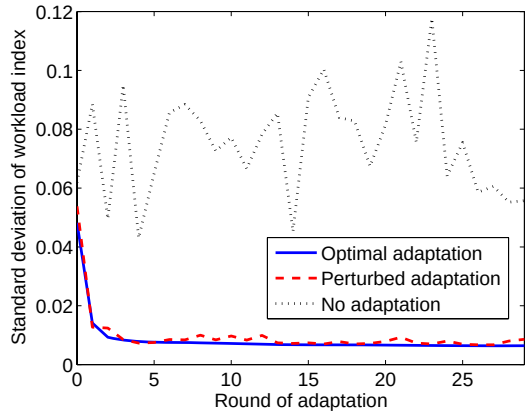


**Figure 75:** Convergence of the standard deviation of workload index in adaptation, plotted by round of adaptation

Figure 74, Figure 75, and 76 plot the experiment results. The dashed line represents the measured results of the "rainy" scenario while the solid line represents the result of the "sunny" scenario. Under both setups, the workload distribution of GeoGrid system converges in the first a few rounds of adaptations. After that, the whole system is stable enough to accommodate the constant moving hot spots without being overloaded. The dotted line represents the performance of a GeoGrid system with no load balance adaptation

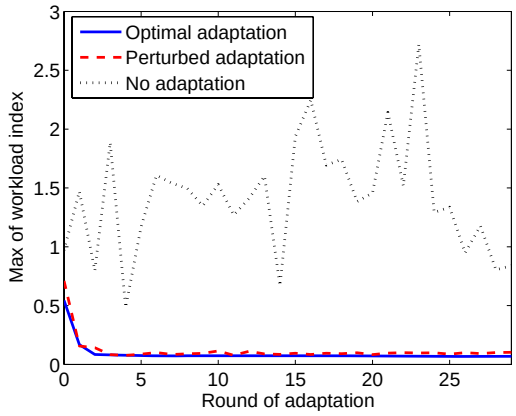


**Figure 76:** Convergence of the max workload index in adaptation, plotted by round of adaptation

mechanism under "rainy" scenario. Compared to the number of GeoGrid system featured with load balance adaptation mechanisms, we can see that the adaptation greatly improved the system load balance.

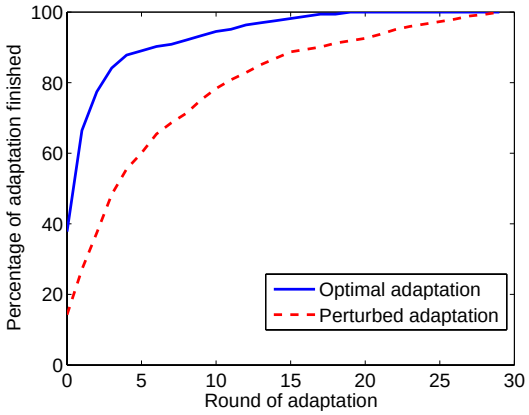


**Figure 77:** Number of load balance adaptations in each round

In the next set of experiment, we record the value of load balance metric at the end of each adaptation. Figure 77 plots the accumulative percentage of load balance adaptations conducted before the end of each round of adaptation. We can see that under "rainy" scenario, more adaptations are needed. It is because the constantly moving hot spots created new overloaded regions when they move into a set of new locations. The number if adaptations conducted in each round decreases, no matter in "rainy" or "sunny" scenarios. This trend shows that the node-region assignment of GeoGrid system converges.

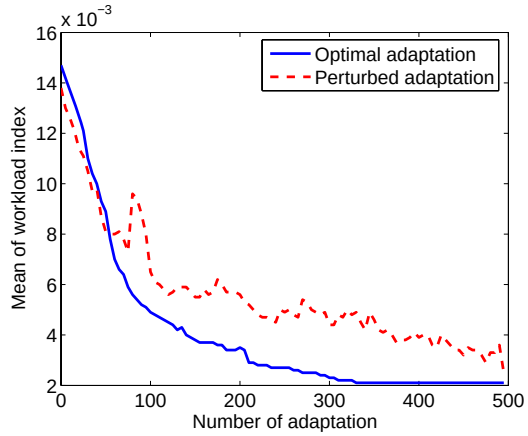


**Figure 78:** Convergence of the mean workload index in adaptation, plotted by number of adaptation

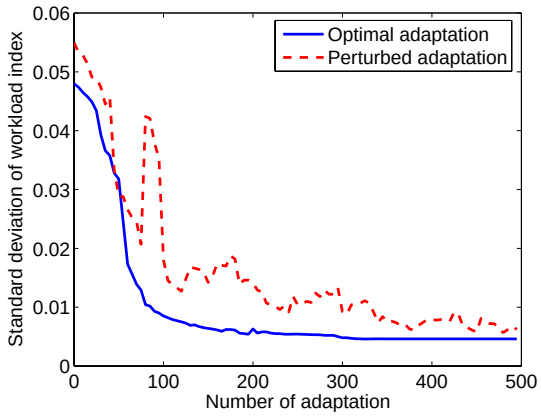


**Figure 79:** Convergence of the standard deviation of workload index in adaptation, plotted by number of adaptation

Figure 78, Figure 79, and Figure 80 plot the recorded load balance measurement results. While the lines of "sunny" scenario converge quickly, the ones of adaptation under "rainy" scenario converges slower than in the figures plotted by the total number of adaptations. In the middle, there are a few surges on the dashed lines, which are caused by the hot spots that move to new locations in between of the first a few adaptation rounds. After a few rounds of adaptations, the whole system converges and the migrating of hot spots is handled more gracefully by GeoGrid systems.

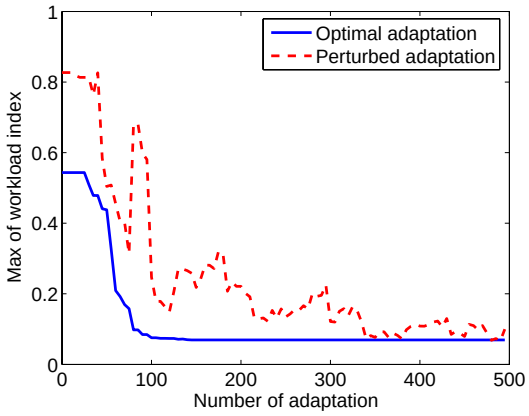


**Figure 80:** Convergence of the max workload index in adaptation, plotted by number of adaptation

## *4.7 Related Works*

A number of research works have been proposed to use a network of caches or an overlay network of end-systems to serve location based applications or mobile users [25, 34]. None of them studied the problem of optimizing the topology of the service network. As our experimental studies suggest, significant improvement on message forwarding efficiency and load balance can be achieved through using the mechanisms proposed in this chapter. Our research work is orthogonal to those works and can be used to improve their performance.

Our solution for improving routing efficiency in GeoGrid network is inspired by the work of Expressway [59] and ECAN [60] system. They proposed similar idea of using shortcuts to accelerate routing in multi-dimensional identifier space. Our system is different from their work in the way we improve the routing workload balance. They considered only the network distance as the metric in building expressway. Their solution can deliver end-to-end routing latency comparable to that of unicast links. However, their solutions did not put node heterogeneity into consideration. In a heterogeneous environment like P2P network, such a design may leads to uneven distribution of workload. Our solution is based on selective random walks. By recruiting more power nodes for processing and forwarding messages, our mechanism can significantly improve the routing load balance.

Research study on CAN network like [45, 53] have proposed a number of solutions for load balance in multidimensional P2P networks. Their solutions are focused on modifying

the bootstrapping process of CAN to smartly choose regions to split. Those solutions may be able to improve the load balance of a network of with more static workload distribution. However, service networks that have dynamically migrating hot spots like GeoGrid, more flexible and responsive load balance mechanisms are needed to improve the system workload distribution. The mechanisms proposed in Section 4.4 can help achieving this goal by dynamically adjust the node distribution to accommodate the changes in workload distribution.

In the database literature, a number of research works have been proposed for improving the workload balance of distributed databases. The basic solution is to map the multidimensional data space into entities of one-dimension space, and dynamically adjusts the item distribution among distributed databases [28, 27]. Whereas such solutions are more suitable for processing range queries in one-dimensional space, they need tremendous modification to support generic location-based applications as GeoGrid could.

Research works like Rebeca [41] and [36] are focused on efficient location-based query processing. Their solutions are based either on centralized services or distributed grid, and did not address the issue of load balance and query routing. GeoGrid is designed as a generic service network, and thus can support the techniques proposed in [41, 36] and add important features like load balance to those solutions.

Other types of networks that can support geographical information dissemination and sharing include Ad-hoc networks [39, 38], Sensor networks [8], and vehicular networks [37]. They usually use multi-hop wireless connections to implement IP unicast features among mobile or fixed nodes, and lack the fixed infrastructures like the GeoGrid service network we present in this chapter. Therefore, information dissemination in these networks has to rely on the unreliable wireless connections, and usually by exploiting the broadcast nature of wireless network links. Because of the limits on the range of wireless signal, it is hard to establish long distance shortcuts like the ones used in GeoGrid. Thus, the end-to-end routing in those networks are less efficient. Furthermore, the physical location and the range of wireless signal determine the location that a node can take in those networks. Dynamic load balance techniques proposed in Section 4.4 can hardly be implemented in

those systems.

# CHAPTER V

# CONCLUSION AND FUTURE WORK

With the rapid advances of wireless communication technology and the increasing popularity of hand-held devices, we witness the continuous escalation of multi-party group communication applications. The implementations of these applications usually involve the exchanges of text or multimedia contents among multiple participants. Traditional client/server architecture can hardly solve the problem because the server side become the bottleneck as the whole system scales, and the end users have to rely on the server for the features they ask for. Future information dissemination applications call for a generic, flexible, efficient, and reliable platform for implementation.

## *5.1 Conclusion*

This thesis aims at developing system-level architectures and techniques to support information dissemination to large scale user groups by using structured P2P overlays, unstructured P2P overlays, and P2P overlays constructed using geographical location information. Although different overlay networks require different system designs for building scalable and efficient information dissemination services, we have employed three common design philosophies: i) exploiting end-system heterogeneity. ii) utilizing proximity information of end-system nodes to localize the communication traffic. iii) using randomized shortcuts to accelerate message routing.

We have demonstrated our design philosophies and the performance improvements in all three types of P2P overlay networks. By statically and dynamically assigning more workloads to more powerful peers, we can greatly increase the system scalability and reduce the variation of workload distribution. By clustering end-system nodes by their IP-network proximity or their geographical proximity, and utilizing randomized shortcuts, we can reduce the end-to-end communication latency, balance peer workloads against service request hot

spots across the overlay network, and significantly enhance the scalability and efficiency of large scale distributed information dissemination and decentralized multi-party group communication.

Our contribution can be summarized by the implementation P2P overlay platforms as following.

**Structured P2P Overlay Network** Designing and implementing an efficient and reliable End System Multicast (ESM) service on top of highly dynamic overlay networks poses several research challenges. In this chapter, we have presented Cascade, an efficient and self-configurable end system multicast system. Our approach has three unique features compared to existing approaches to application-level multicast systems.

- First, we use the landmark signature technique to cluster end-hosts in the Cascade ESM overlay network, aiming at exploiting the network proximity of end-hosts for efficient multicast information dissemination across wide area networks.
- Second, we propose a capacity-aware overlay construction technique based on the concept of virtual node to balance the multicast workload among heterogeneous end-hosts.
- Third, we develop a dynamic passive replication scheme to provide reliable ESM services in an environment of inherently unreliable peers. An analytical model is presented to discuss its fault tolerance properties.

We evaluate Cascade using simulations of large scale networks. The experimental results indicate that Cascade can provide efficient and scalable multicast services over large-scale network of heterogeneous end-system nodes, with reasonable link stress and good load balance.

**Unstructured P2P Overlay Network** We have described Peercast, a wide-area group communication system, focusing on the design and evaluation of the utility-based distributed algorithms for managing the overlay topology and constructing the information

dissemination spanning trees. This chapter makes three unique contributions.

- First, we identify network proximity and node capacity as two utility metrics and show that a careful combination of these two metrics can have significant impact on the efficiency and scalability of wide-area group communications.
- Second, we introduce a utility-aware P2P overlay management protocol in the bootstrapping process.
- Third but not the least, we describe the design of a utility-aware wide-area group communication management protocol that is capable of dynamically combining network proximity and node capacity to deliver considerably improved performance for wide-area group communications.

Our initial experimental results show that Peercast can improve the scalability of wide-area group communications by one to two orders of magnitude.

**Location-based Service Network** In this chapter, we presented GeoGrid system, a system that organizes end-systems into a P2P network of proxies for mobile users. GeoGrid system is unique in the following aspects:

- It is featured with the management and routing mechanisms optimized for handling location-based information dissemination and sharing.
- Its dynamic load balance scheme can distribute the information processing and dissemination workloads according to the geographical distribution and capacity of end-systems.
- The GeoGrid system is free from the control of service providers. End users and open source community can leverage the generic communication interfaces provided by the GeoGrid system to support various applications and to provide location-based services.
- Furthermore, GeoGrid can also be used to manage the server farms of service providers and provide geographical information services.

## 5.2 Future Work

We have addressed the design issues of scalability, reliability, and efficiency in our study of information dissemination to large scale user groups. However, we have left a number of research topics for our future studies, even though those topics are interesting complementary of the works presented in this thesis. We record them below, as a memorandum of directions for our future researches.

**Security** We design our system as an open platform on which security mechanisms like Event Guards [51] could be implemented. Such solutions usually improve the system security by adding cryptographical mechanisms, which will consequentially increase the communication and computation overhead in the system. Our proposed solutions increase the system efficiency of message passing, and will help reduce the performance degradation caused by those security mechanisms.

**Trust Management** It will be an interesting topic to consider trust management in our systems. Because of the highly dynamic nature of P2P network nodes, the information dissemination services will not be stable if it relies on those nodes with high turn over rates for services. Although our service replication mechanisms of Chapter 2 can increase the availability of information dissemination services, handover and buffer delay on end system nodes will still have negative impacts on the application quality of services. If we could incorporate trust management mechanisms into the overlay networks, and mapping the availability of individual nodes to their trustworthiness, we can improve the over-all system availability by putting more information dissemination workloads on more reliable nodes that have more stable network connections.

**Hybrid Topology** Our system does not exclude the architecture of "supernode" or multi-layer P2P networks. Instead, our scheme can be used to construct the higher layer overlay that interconnects supernodes. Normal peers can simplify their bootstrapping process by connecting only to supernodes. Our utility-aware peer selection mechanism of Chapter 3 can help peers to identify the supernodes that are close to them, such that the communication

between a leave peer and its supernode can be minimized.

# REFERENCES


[1] "Computer and Internet use in the United States: 2003." http://www.census.gov/prod/2005pubs/p23-208.pdf, July 2006.

[2] "E. Ng. GNP software." http://www.cs.rice.edu/ eugeneng/research/gnp/software.html, July 2006.

[3] "Google Talk." www.google.com/talk/, July 2006.

[4] "Live Meeting." http://www.microsoft.com/office/livemeeting, July 2006.

[5] "Sharman networks LTD. KaZaA media desktop." http://www.kazaa.com, July 2006.

[6] "Skype." http://www.skype.com, July 2006.

[7] Aberer, K., Cudr-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., and Schmidt, R., "P-grid: A self-organizing structured p2p system," *SIGMOD Record*, vol. 32, pp. 29 – 33, September 2003.

[8] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E., "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[9] Alsberg, P. and Day, J., "A principle for resilient sharing of distributed resources," in *Proceedings of ICSE*, 1976.

[10] Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R., "Resillient overlay networks," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, 2001.

[11] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., and Khuller, S., "Construction of an efficient overlay multicast infrastructure for real-time applications," in *Proceedings of INFOCOM*, 2003.

[12] Banerjee, S., Bhattacharjee, B., and Kommareddy, C., "Scalable application layer multicast," in *Proceedings of the 2002 ACM SIGCOMM Conference*, 2002.

[13] Baset, S. A. and Schulzrinne, H., "An analysis of the skype peer-to-peer internet telephony protocol," Tech. Rep. cucs-039-04, Department of Computer Science, Columbia University, September 2004.

[14] Bu, T. and Towsley, D., "On distinguishing between internet power law topology generators," in *IEEE INFOCOM*, (New York, NY), IEEE, June 2002.

[15] Carzaniga, A. and et al., "A routing scheme for content-based networking," in *INFOCOM*, 2004.

[16] Carzaniga, A., Rosenblum, D. S., and Wolf, A. L., "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, 2001.

[17] Castro, M., Druschel, P., Kermarrec, A., and Rowstron, A., "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in communications (JSAC)*, 2002.

[18] Chawathe, Y., *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California, Berkeley, 2000.

[19] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S., "Making Gnutella-like P2P systems scalable," in *ACM SIGCOMM*, (Karlsruhe, Germany), ACM, August 2003.

[20] Chu, Y.-H., Rao, S. G., and Zhang, H., "A case for end system multicast," in *ACM SIGMETRICS 2000*, pp. 1–12, ACM, 2000.

[21] Dabek, F., Cox, R., Kaashoek, F., and Morris, R., "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM*, (Portland, Oregon, USA), ACM, August 2004.

[22] Deering, S. and Cheriton, D., "Multicast routing in datagram internetworks and extended lans," *ACM Transactions on Computer Systems*, vol. 8, May 1990.

[23] Deering, S., Estrin, D., Farinacci, D., Jacobson, V., Liu, C., and Wei, L., "The PIM architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, April 1996.

[24] Faloutsos, M., Faloutsos, P., and Faloutsos, C., "On power-law relationships of the internet topology," in *ACM SIGCOMM*, pp. 251–262, ACM, 1999.

[25] Fiege, L., Gartner, F. C., Kasten, O., and Zeidler, A., "Supporting mobility in content-based publish/subscribe middleware," in *Middleware*, 2003.

[26] Francis, P., "Yoid: Extending the multicast internet architecture." 1999.

[27] Ganesan, P., Bawa, M., and Garcia-Molina, H., "Online balancing of range-partitioned data with applications to peer-to-peer systems," in *VLDB*, 2004.

[28] Ganesan, P., Yang, B., and Molina, H. G., "One torus to rule them all: Multi-dimensional queries in p2p systems," in *WebDB*, 2004.

[29] Gedik, B. and Liu, L., "Reliable peer-to-peer information monitoring through replication," in *Proceedings of IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2003.

[30] Gedik, B. and Liu, L., "Mobieyes: A distributed location monitoring service using moving location queries," *IEEE Transactions on Mobile Computing*, 2005.

[31] Gedik, B. and Liu, L., "A scalable peer-to-peer architecture for distributed information monitoring applications," *IEEE Transactions on Computers*, vol. 54, no. 6, pp. 767 – 782, 2005.

[32] Guerraoui, R. and Shiper, A., "Software-based replication for fault tolerance," *IEEE Computer - Special Issue on Fault Tolerance*, vol. 30, pp. 68–74, 1997.

[33] Gueye, B., Ziviani, A., Crovella, M., and Fdida, S., "Constraint-based geolocation of internet hosts," in *Proceedings of Internet Measurement Conference*, pp. 288–293, 2004.

[34] Huang, Y. and Garcia-Molina, H., "Publish/subscribe in a mobile environment," in *MobiDE*, 2001.

[35] Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., and O'Toole, Jr., J. W., "Overcast: Reliable multicasting with an overlay network," in *Proceedings of Symposium on Operating System Design and Implementation (OSDI)*, pp. 197–212, 2000.

[36] Liu, B., Lee, W.-C., and Lee, D. L., "Supporting complex multi-dimensional queries in p2p systems," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, 2005.

[37] Luo, J. and Hubaux, J.-P., "A survey of inter-vehicle communication," Tech. Rep. IC/2004/24, School of Computer and Communication Sciences, EPFL, 2004.

[38] Maihöfer, C., "A survey on geocast routing protocols," *IEEE Communications Surveys and Tutorials*, vol. 6, no. 2, 2004.

[39] Mauve, M., Widmer, J., and Hartenstein, H., "A survey on position-based routing in mobile ad hoc networks," *IEEE Network Magazine*, vol. 15, pp. 30–39, November 2001.

[40] Merugu, S. and et al, "p-sim: A simulator for peer-to-peer networks," in *MASCOTS*, 2003.

[41] Muhl, G., Ulbrich, A., Herrmann, K., and Weis, T., "Disseminating information to mobile clients using publish-subscribe," *IEEE Internet Computing*, vol. 08, no. 3, pp. 46–53, 2004.

[42] Palmer, C. R. and Steffan, J. G., "Generating network topologies that obey power laws," in *Proceedings of GLOBECOM '2000*.

[43] Pandurangan, G., Raghavan, P., and Upfal, E., "Building low-diameter p2p networks," in *Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science*, 2001.

[44] Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M., "ALMI: An application level multicast infrastructure," in *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, pp. 49–60, 2001.

[45] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., "A scalable content addressable network," in *Proceedings of SIGCOMM*, ACM, 2001.

[46] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., "Application-level multicast using content-addressable networks," *Lecture Notes in Computer Science*, vol. 2233, 2001.

[47] Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., "Topologically-aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM*, 2002.

[48] Ripeanu, M., Foster, I., and Iamnitchi, A., "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing Journal*, vol. 6, no. 1, 2002.

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

[50] Saroiu, S., Gummadi, P., and Gribble, S., "A measurement study of Peer-to-Peer file sharing systems," in *Proceedings of MMCN*, (San Jose, CA), August 2002.

[51] Srivatsa, M. and Liu, L., "Securing Publish-Subscribe overlay services with EventGuard," in *Proceedings of ACM Computer and Communication Security (CCS 2005)*, 2005.

[52] Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H., "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proceedings SIGCOMM*, pp. 149–160, ACM, 2001.

[53] Takemoto, D., Tagashira, S., and Fujita, S., "Distributed algorithms for balanced zone partitioning in content-addressable networks," in *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS'04)*, p. 377, 2004.

[54] Tang, L. and Crovella, M., "Virtual landmarks for the internet," in *Internet Measurement Conference*, pp. 143 – 152, 2003.

[55] Watts, D. J. and Strogatz, S. H., "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[56] Wouhaybi, R. H. and Campbell, A. T., "Phenix: Supporting resilient low-diameter peer-to-peer topologies," in *IEEE INFOCOM*, (Hong Kong, China), IEEE, March 2004.

[57] WWW, "Gnucleus. The Gnutella web caching system," *http://gnucleus.sourceforge.net*, July 2006.

[58] WWW, "The Gnutella RFC," *http://rfc-gnutella.sourceforge.net*, July 2006.

[59] Xu, Z., Mahalingam, M., and Karlsson, M., "Turning heterogeneity into an advantage in overlay routing," in *Proceedings of INFOCOM*, 2003.

[60] Xu, Z., Tang, C., and Zhang, Z., "Building topology-aware overlays using global soft-state," in *Proceedings of ICDCS*, 2003.

[61] Zegura, E. W., Calvert, K. L., and Bhattacharjee, S., "How to model an internetwork," in *IEEE Infocom*, vol. 2, pp. 594–602, IEEE, March 1996.

[62] Zhang, J., Liu, L., and Pu, C., "Contructing a proximity-aware power law overlay network," in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM 2005*.

[63] Zhang, J., Liu, L., Pu, C., and Ammar, M., "Reliable peer-to-peer end system multicasting through replication," in *Proceedings of IEEE P2P*, 2004.

[64] Zhao, B., Kubiatowicz, J., and Joseph, A., "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," tech. rep., U. C. Berkeley, 2002.

[65] Zhao, B. Y., Duan, Y., Huang, L., Joseph, A. D., and Kubiatowicz, J. D., "Brocade: Landmark routing on overlay networks," in *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*.

[66] Zhou, Y. and et al., "Adaptive reorganization of conherency-preserving dissemination tree for streaming data," in *ICDE*, 2006.

[67] Zhuang, S., Zhao, B., Joseph, A., Katz, R., and Kubiatowicz, J., "Bayeux: An architecture for scalable and fault-tolerant widearea data dissemination," in *Proc. of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001)*, 2001.

# VITA

Jianjun Zhang was born and raised in Wuhan, a city known as "great river city", in China. He obtained a bachelor degree (B.S.) in Computer Science and a master degree (M.E) of Computer Organization and Architecture, from the Computer Science department of Wuhan University in 1996 and 1999 (Wuhan, China). It is the most exciting seven years in his life, because he met his wife Han Zheng on the beautiful campus of Wuhan University. After his graduation from Wuhan University, he joined Bell Labs (Beijing) of Lucent Technologies (China) and worked there as a software and system test engineer till the summer of 2001. Subsequently, he joined Computer Science Ph.D. program at the College of Computing of Georgia Institute of Technology (Atlanta, GA, USA). As a member of the DiSL research group at the College of Computing, he conducted research on various aspects of distributed data intensive systems, including peer-to-peer computing, end-system multicast, web information retrieval and integration, and web services integration. He led three projects in the DiSL research group, namely Cascade, PeerCast, and GeoGrid. He also helped deliver project XWRAPComposer and LDRD. His research in these projects has resulted in a number of publications that have appeared in various journals and international conferences on network and data engineering.