# UNIVERSITY OF CINCINNATI

**Date:** Jan 25, 2008

**I,** Honghao Wang **,**

**hereby submit this work as part of the requirements for the degree of:**

Doctorate of Philosophy

**in:**

Computer Science and Engineering

**It is entitled:**

An Efficient and Secure Overlay Network for General Peer-to-Peer Systems

**This work and its defense approved by:**

**Chair:** Dr. Yiming Hu

Dr. Dharma Agrawal

Dr. Carla Purdy

Dr. Karen Tomko

Dr. Lin Liu

# An Efficient and Secure Overlay Network for General Peer-to-Peer Systems

by

Honghao Wang

M.S. Institute of Software, Chinese Academy of Sciences 2000
B.S. Huazhong University of Science and Technology 1997

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science and Engineering

in the

Department of Electrical and Computer Engineering and Computer Science
of the
College of Engineering
of the
University of Cincinnati, Ohio

**Committee:**
Professor Yiming Hu, Chair
Professor Dharma P Agrawal
Professor Carla Purdy
Professor Karen Tomko
Professor Lin Liu

Winter 2008

## Abstract

Currently, Peer-to-Peer overlays can be classified into two main categories: unstructured and structured ones. Unstructured overlays are simple, robust, and powerful in keyword search. Structured ones can scale to very large systems in terms of node number and geography, and guarantee to locate an object within $O(LogN)$ hops. However, both of them face difficulties in efficiency and security of overlays. For unstructured ones, the efficiency problem presented is poor scalability. For structured ones, it is long routing latency and enormous overhead on handling system churn. Moreover, both of them are vulnerable to malicious attacks.

Peer-to-Peer overlays belong to application-level network. To a great extension, overlay network designs ignore physical characteristics. As the result, their structures are far from underlying physical network or the distribution pattern of overlay peers. These inconsistencies induce system common operations costly, such as routing and lookup. On the other hand, most peers are assumed to have uniform resources and similar behaviors. Thus, Peer-to-Peer protocols were designed to be symmetric. However, in the realistic environment, peers' resources and behaviors are highly skewed. Symmetric protocols actually compromise system performance. Frequently joining and leaving of peers generates enormous traffic. The significant fraction of peers with high latency/low bandwidth links increase lookup latency. Moreover, under the environment without mutual trust, Peer-to-Peer systems are very vulnerable for varied attacks because they lack a practical authentication mechanism.

From a different perspective, this dissertation proposes to construct a highly efficient and secure Peer-to-Peer overlay based on the physical network structure of the Internet and network locality of overlay peers. By naturally integrating different network-aware

techniques into the Peer-to-Peer overlay, a novel SNSA (Scalable Network Structure Aware) technique has been developed. It can provide accurate information of network locality of overlay peers and sufficient physical network structure of the Internet. Based on the valuable information, a unique Peer-to-Peer overlay, which can reflect network structure and locality of overlay peers, is constructed. Also, peers are assigned different roles by their resources and behaviors. Minor capable peers are involved in overlay core operations, such as routing and lookup. Major normal ones are organized into highly dependable teams, and assigned usual tasks, such as storing objects. Not only can this overlay support both structured and unstructured systems, but also the systems are highly efficient in routing and consuming much less bandwidth.

As the observation that every peer must subject to the network configuration and administration imposed by ISPs, we propose to identify each peer by its physical network characteristic, *net-print*. Based on the SNSA technique and the net-print, a distributed authentication and secure routing mechanisms are developed under Peer-to-Peer environment.

Beware of the fact that every overlay network maintains its own network proximity system. This dissertation proposes to build a common layer to provide such information for all overlays. By deeply analyzing requirements of current overlays, three kinds of primitives are designed to provide valued knowledge of physical network and overlay peers. Not only dose this method save network resource by eliminating duplicated probes, but it also provides an efficient way to share information between overlays.

*Keywords*—Peer-to-Peer, Overlay Network, Overlay Routing, Overlay Structure, Distributed Hash Table (DHT), Network Topology, Network Aware, Network Locality, Network Proximity, Network Security, Secure Routing, System Churn

**Dedication**

To my parents and dear wife,

to all people I love.

## Acknowledgments

I would like to thank my advisor Dr. Yiming Hu who provided constant guidance and support to explore new areas of Peer-to-Peer. During my whole PhD study, he has always instilled in me innumerable lessons and insights on the academic research and professional developments.

I would also like to extend my sincere appreciation to my committee members, Dr. Dharma Agrawal, Dr. Carla Purdy, Dr. Karen Tomko, and Dr. Lin Liu, for reviewing previous draft of this dissertation and providing many valuable comments that improve the presentation and contents of this dissertation.

Last but not the least, I would like to express my earnest gratitude to my parents for their love and support. From the bottom of my heart, my thanks go to my dear wife Lin Jiang, who is always my most precious one to share the life with. Your love, support and help are indispensable. Thanks to my daughter Sophia for joy, happiness and big smile she brings to me.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Peer-to-Peer Systems

The last few years have seen a tremendous increase in the interest and research activities of Peer-to-Peer (P2P) overlays. While P2P research covers a wide spectrum of topics, such as routing/lookup, security, file systems, load-balancing, etc, one of the most fundamental research topics is how to provide an efficient and secure lookup service in a large-scale network which is completely distributed and decentralized.

The primary goal of the P2P system is to harness and aggregate the slack resources(e.g., bandwidth, storage and CPU) on each peer to build large-scale distributed applications, such as wide-area file sharing, distributed web cache, and distributed file/storage systems. Generally, P2P systems can be categorized into two major architectures: unstructured and structured. The first architecture is recognized as the first generation of P2P systems, while the second one is recognized as the second generation.

### 1.1.1 Unstructured P2P Systems

For unstructured P2P systems, such as Gnutella [2] and KaZaA [3], their overlay topology can be viewed as a random graph where each node randomly chooses some nodes as neighbor nodes. Gnutella is a typical example of the unstructured P2P network. In Gnutella, the peers form an overlay network by forging point-to-point connections with a set of neighbors, and each one maintains the state of its neighbors. To deal with peer joins and departures, Gnutella uses ping and pong messages to maintain the overlay network.

To locate a document, a peer initiates a controlled flooding by sending a query to all of its neighbors. Upon receiving a query, the peer checks if any locally stored documents match the query. If so, the peer sends a query response back towards the query originator. Whether or not a document match is found, the peer continues to flood the query until the TTL (time-to-live) reaches 0. Upon receiving a query response, the query originator may initiate a file download directly from the peer which gives the query response.

Although the overlay structure is disordered, it is simple to construct and robust in face of major disasters. It can keep peers highly connected even when a large amount of peers suddenly leave the overlay. When performing searches on arbitrary overlays, it is hard to be efficient. The flooding mechanism is currently used for unstructured search. Although such mechanism can make the query reach most of its host within several forwards, it involves most of peers per query and generates a large amount of duplicate messages, which is far from efficient and causes a scalability problem.

Based on the proprietary Fasttrack [3] technology that uses a special *supernodes* design, KaZaA becomes popular. Those supernodes always have higher network bandwidth and connect with hundreds of peers, which make them good collectors to cache published information of neighbors. Queries are first routed to supernodes. Normally, queries for prevalent objects, such as popular music files, can be quickly solved by accessing several

supernodes. Although those supernodes may become hot points or bottlenecks, they improve searching efficiency and scalability compared with original unstructured designs. The similar technique is also adopted by the Gnutella2 [4], which is the next generation of Gnutella.

### 1.1.2 Structured P2P Systems

Although unstructured P2P systems are simple, robust and able to locate popular objects, they face difficulties in scalability and locating rare objects. Structured P2P systems based on distributed hash tables (DHTs) have recently attracted tremendous attention from research communities. The representative systems include Chord [5], Pastry [6], Tapestry [7] and CAN [8].

DHTs belong to content-addressable overlay networks. Each node within the network is assigned an ID, which normally is produced by the hashing of its IP address or public key. Also, each document (or piece of content) has a key, produced by the hashing of its name or content. One node manages a range of keys. Compared with unstructured systems, DHTs have strict overlay topology and data placement. Each node's ID determines a node's position in the overlay and the document's key determines which node to store a document. The storage and retrieval of a document is essentially a process of mapping from the document's key to a node's ID.

Within an N-node DHT, each node maintains a routing table with node IDs of other $O(logN)$ nodes and their associated IP addresses. Because the routing table is distributed over nodes, the DHT can resolve all lookups via $O(logN)$ messages to other nodes. For fault-tolerance, documents are stored at multiple nodes in the overlay. DHTs are particularly attractive for the construction of a variety of decentralized services due to their scalability, availability, fault-tolerance and self-organization. DHTs can scale to a large number of nodes (say, millions of nodes) due to their $O(logN)$ routing table size on each node

and $O(logN)$ messages during a lookup. They can achieve availability and faulty-tolerance through data replication and routing link redundancy.

Two representatives of DHTs, Chord and Pastry, will be briefly introduced.

**Chord**

Chord maps a 160-bit key to a set of IP addresses on the nodes responsible for the key. It uses consistent hashing, which has several good properties [5]. With high probability, the hash function not only balances load over nodes, but also moves a minimum load to maintain a balanced load when an $N^{th}$ node joins or leaves the system, i.e., only an $O(1/N)$ fraction of the keys and their associated documents.

Chord uses a 160-bit circular ID space where each node has a 160-bit ID. The $s$ nodes whose IDs immediately follow a key are considered responsible for that key: they are the key's successors. To provide reliable lookup even if half of the nodes fail in a $2^{16}$ node network, the number of successors, $s$, is 16 in the Chord implementation. The ID space in Chord wraps around such that zero immediately follows $2^{16}$-1.

The base Chord lookup algorithm works as follows. Each Chord node (say $x$) maintains a routing table, which includes a *finger table* and a *successor list*. The finger table consists of the IP addresses and IDs of nodes which follow the Chord node $x$ at power-of-two distances in the identifier space (i.e., $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, ...). The successor list refers to $x$'s immediate successors. When a node issues a lookup with a key $k$, it consults a sequence of other nodes, asking each in turn which node to talk to next. Each node in this sequence answers with the node from its finger table whose ID most immediately precedes $k$. By $O(logN)$ consultations, the originating node will find the key k's predecessor node, and then it requests the predecessor node for its successor list, which is the result of the lookup. Please refer to [5] for more details of Chord.

**Pastry**

Pastry node IDs are assigned randomly with uniform distribution from a circular 128-bit ID space [6]. It supports the *lookup(key)* operation which maps a 128-bit key to the node whose ID is numerically closest to the key. Both node IDs and keys can be thought of as a sequence of digits in base $2^b$.

Each node maintains a *routing table* and a *leaf set*. A node's routing table is made up of $log_{2^b}N$ rows with $2^b$-1 columns each ($b$ is a configuration parameter with a typical value of 4). The $2^b$-1 columns at row $r$ each contain the IP address of the node whose ID shares the present node's ID in the first $r$ digits, but whose $r+1$th digit has one of the $2^b$-1 possible values other than the $r+1$th digit in the present node's ID.

Pastry's routing algorithm is a prefix-based lookup. Given a message with a key $k$, a node seeks to forward the message to a node in the routing table whose ID shares with the key a prefix that is at least one digit (or $b$ bits) longer than the prefix that the key shares with the present node's ID. If no such node can be found, the message is forwarded to a node whose ID shares a prefix with the key as long as the present node, but is numerically closer to the key than the present node's ID. If no appropriate node exists in either the routing table or leaf set, then the present node or its immediate neighbor is the message's final destination. Please refer to [6] for more detail of Pastry.

Although DHT-based systems guarantee to solve a request within $O(LogN)$ hops, it dose not mean the routing/lookup is efficient. A previous study [9] has shown that a significant fraction of peers could be connected over high latency/low bandwidth links, such as dialup. Thus, the presence of even one such slow logical hop on a logarithmically long path is likely. This increases the overall cost of the routing. Furthermore, routing performance will be worse under system churn. As mentioned by Rhea et al. in [10], for 1,000 nodes system under modest churn rate, 23 minutes of each node's median session time

(many researches have recorded most of session times are less than 5 minutes), a Pastry system (FreePastry) has failed to complete 70% requests. Although almost all lookups in a Chord network have been completed, the latency increased more than 20 times. In order to handle system churn, Bamboo [11] has been proposed. Each node was designed to periodically detect its neighbors within the routing table and share its leaf set with logical nearby nodes. However, the overhead of such maintenance was substantial. The bandwidth consumed under its default period was 1.8kps per node, which would generate 7.5 times more traffic than the unstructured Gnutella system [12].

## 1.2 Inconsistencies Behind P2P Overlays

Two obvious inconsistencies are the major reasons for above difficulties. The first one is the structure inconsistency, which is between the regular overlay structure and either the skewed distribution of overlay peers or the physical network structure of the Internet. The second one is the responsibility inconsistency, which is between diverse resources/behaviors of peers and their uniform responsibility.

### 1.2.1 Structure Inconsistency

Since P2P overlay is an application level network, network transparency masks underlying network details. Current P2P protocols, to a great extent, ignore the underlying physical network. As a result, an ideal overlay structure tends to be used. For example, a ring is used in Chord, and trees are used in Pastry.

However, these structures neither identify the underlying network structure nor match the distribution of peers. Previous research [1] has pointed out that the physical structure of the Internet follows the power-law. That is to say some networks in the Internet connect with many others, and most of networks connect with only very few ones. Moreover,

the distribution of the P2P overlay peers is far from uniform. Previous research [9] has pointed out more than 40% of peers are located within the top ten Autonomous Systems (ASes). As a result, the logical related nodes are normally far from each other physically in both structured and unstructured P2P systems.

Due to the logical locality existing in any distributed application, a small fraction of many host pairs in a system account for most end-to-end network traffic. In the P2P environment, the peers with logical locality are structure related ones, such as the nodes within neighboring table for unstructured protocols, and the ones within the routing table for structured designs. Because of the mismatch between ideal overlay structure and both distribution of peer and underlying physical network, the communication overhead among logical related peers is substantial. Moreover, these communications are the most common cases in the systems. They are always the atomic operations for all P2P functions, such as routing and lookup. Thus, P2P systems are far from optimization due to the structure inconsistency.

Being aware of this problem, many works have been done. For unstructured overlays, network proximity is used to optimize connections within the neighbors of a peer or its neighbors' neighbors by Liu, etc. [13]. For structured ones, network and geography proximity are widely used to optimize overlay construction and selection of the next hop in routing. While those works significantly improve performance, they have limitations. For unstructured protocols, more than 50% of the neighbors of a peer could not be optimized, as considerations of network disasters, such as partition, and deny of service by malicious peers were around. For structured ones, the latest research [14] has pointed out that the latency of the last few hops under Proximity Neighbor Selection (PNS) in a lookup still approximated 1.5 times the average round trip time because of insufficient proximity to its neighbors.

### 1.2.2 Responsibility Inconsistency

By assuming most of the peers in the P2P environment are uniform in resources and behaviors, most of the P2P protocols are symmetric, which grants each node equal responsibility. However, previous researches [12, 15, 9] have pointed out that the distribution of recourses and behaviors of peers are highly skewed. For example, while 58% of peers are connected with high speed connection techniques, such as DSL and cable modem, 31% of peers are still using dial-up. Although most of the peers have very low uptime or permanently leave the system after staying several minutes, there are 10% and 18% nodes with 90% uptime (availability) in Gnutella and Napster, respectively. Moreover, those 10% or 18% nodes contribute about 90% of the total traffic. Although current P2P protocols successfully integrate various peers into an overlay by consuming little resources at each one, to grant each node equal responsibility by ignoring their differences may seriously compromise efficiency and security of an overlay. For example, high latency/low bandwidth nodes increase routing/lookup latency. Frequently joining and leaving of nodes multiply maintenance overhead. Malicious ones threaten overlay security.

As a result, three types of peers are considered inappropriate to be involved in overlay core operations, such as routing and lookup, especially for structured overlays. The first one is *transitory peers*, which have short *lifetime* or low *availability*. A node's *lifetime* is the time between when it enters the overlay for the first time and when it leaves the overlay permanently. A node's *session time* is the elapsed time between when it joins the overlay and when it subsequently leaves the overlay. The sum of a node's session times divided by its lifetime is defined as its *uptime* or called *availability*. The second is *weak peers*, which connect with high latency/low bandwidth techniques, such as dial-up. The third kind is *malicious peers*.

Since P2P overlays tend to grant each peer equal responsibility, the work of lookup

and storing objects are uniformly distributed among peers in order to balance system load. In other words, the possibility of each peer being accessed (each node's contribution to the overlay) is not high. Thus, the initial overhead, such as building and updating routing tables and moving datum, becomes substantial for transitory peers. Assuming a Chord overlay with $N$ nodes and $K$ objects is stored, building the routing table of a joining node will involve $(logN + log^2N)$ unit of traffic, which includes the joining lookup and building the routing table from passing nodes; to notify related nodes updating their routing tables costs $log^2N$ unit, which includes $logN$ notice lookups; to move objects consumes $K/N$ unit. Thus, the overhead of a node's joining $Cost_{init}$ is $(logN + 2\,log^2N + K/N)$. Assuming that the average node request rate is $\gamma$ and lifetime is $T_{life}$, the useful traffic $Cost_{useful}$ is $(logN\ \gamma\ T_{life})$. Assuming $N$ is one million, $\gamma$ is 0.1 requests per node per second and $K/N$ is 300, the $T_{life}$ should be at least 20 minutes to make $Cost_{useful}$ equal to $Cost_{init}$. Considering lots of nodes permanently leave the system after several minutes, setting up datum in those may significantly increase the overhead of the whole overlay.

While the short lifetime nodes raise the overhead of overlays, the nodes with low availability and high latency/low bandwidth links compromise system performance. We assume a bound $f$ on the fraction of nodes within the routing table that are unavailable. The fraction of those nodes among the overlay is also considered $f$, since P2P protocols treat each node equally. The average latency of each hop is $L$, and to timeout an unavailable node costs $5L$. The total latency for a lookup in Chord is $\frac{(L\,(1-f)+5\,L\,f)\,logN}{1-f}$. Considering a representative median availability of 30% of current P2P systems, the total latency is increased 12.7 times. The analysis is similar involving peers with high latency/low bandwidth links. With 31% dial-up nodes included, the total latency is increased 2.2 times. Considering all together, the total latency will be increased 20 to 30 times. It significantly impacts overlay performance.

Although current P2P designs can tolerate some fault of nodes, the existence of

malicious nodes is a serious threat. For structured overlays, by controlling a bound of node-IDs, malicious nodes can threaten overlay routing or even block/shade some target nodes. While malicious nodes tend to be a little fraction, with the total number of nodes $N$ increasing, their infection can be amplified. A lookup may fail if any of the nodes along the route are faulty; faulty nodes may simply drop the message, route the message to the wrong place or pretend to be the key's root. Therefore, the probability of a successfully lookup when a fraction of $f$ of the node is faulty is only $(1-f)^{logN}$. In order to affect 10% of the routing/lookup, the $f$ is 2.6% for an overlay with $10^4$ nodes, but the number reduces to 1.7% for $10^6$ nodes. Moreover, malicious nodes can easily increase their number by presenting multiple node-IDs, which is called a *Sybil attack*; or collaborate with each other, which is called *collusion*. Actually, those attacks can seriously compromise structured overlays, and have raised much research [16, 17, 18]. However, due to the complexity of the P2P environment, many solutions are not satisfied, and some issues are still open, such as Sybil attack and collusion of malicious nodes under dynamic IP environment. For unstructured overlays, because malicious nodes normally have higher network bandwidth, longer uptime, more powerful CPU and more disk space, they are always suitable neighbors/supernodes, and have many more connections than normal ones. As a result, it is easier for them to affect other nodes in the overlay.

## 1.3 Solutions to Inconsistencies

Our solution to deal with Structure Inconsistency is to build a P2P overlay by exploiting the physical network structure of the Internet and the network locality of overlay peers simultaneously[1]. Compared with current P2P designs and network-aware adaptations, our solution has two major differences. Due to network transparency consideration,

[1]To our best knowledge, it is our first to propose building P2P overlays based on physical network structure of the Internet and network locality of overlay peers.

current P2P designs employ top-down method, and use network-aware information as a supplemental method. However, the bottom-up method is adopted in our overlay design. The overlay structure of our design essentially bases itself on physical network of the Internet and the locality of overlay peers. In another word, to a great extent, our overlay directly reflects the physical network structure and the locality of overlay peers, while others use network-aware information to improve their overlays. Secondly, current designs normally exploit round-trip-time (RTT) to acquire network-aware information. While RTT is easy to measure and suitable for a distributed environment, it has limitations such as stability problem and insufficient ability to reflect network structure and locality. Based on our novel Scalable Network Structure Aware (SNSA) technique, the physical network structure and locality of peers are accurately acquired and naturally exploited to build the overlay. Based on this overlay, which directly reflects network structure and locality of peers, characteristics of the physical network, such as the power law of the Internet and major asymmetric network connections, are naturally utilized, network locality of peers is exploited maximally, and the common/atomic operations of systems are optimized.

Our solution to deal with responsibility inconsistency is to grant each peer a different responsibility related to its behavior and resource. Instead of involving all kinds of nodes into overlay core operations, such as routing and lookup, only qualified nodes will be assigned. The maintenance overhead of an overlay $Cost_{maintn}$ should be a direct proportion function of the number of peers involved, $N$, and their changing frequency $R$. By excluding transitory nodes, both $N$ and $R$ are significantly decreased. As a result, $Cost_{maintn}$ reduces dramatically. Moreover, the lookup latency is decreased, since both the number of hops and the latency of each hop are reduced.

Although normal and incompetent peers are not suitable to be involved in core operations, their number is huge and their resources are valuable. However, to harness the resources is not easy, since those nodes normally have lower availability and/or network

bandwidth. Thus, a highly dependable and efficient protocol is needed in order to exploit their resources. Since routing operations are performed by core nodes, another major work of P2P overlays is to store objects. Thus, we plan to exploit those nodes to form a highly dependable storage. The *erasure code* technique becomes our solution. By striping an object into coding blocks and distributing among nodes, erasure code can provide very high dependability even when the dependability of each node is low. By configuring coding parameters, an object can be rebuilt by retrieving any of $n$ blocks from $m$ nodes, in which $n$ is less than $m$. Erasure code not only archives dependability, but it also provides efficiency. Major Internet connections of peers are either asymmetric ones, such as DSL and cable modem, or dial-up one. Their common characteristic is that the uploading bandwidth is normally 1/8 to 1/13 of the downloading one. Thus, to retrieve an object from many nodes with the erasure coding technique is much more efficient. In summary, granting nodes suitable works based on their behaviors and their resource significantly improves dependability and efficiency of the whole overlay.

## 1.4 Security Issues

To provide secure routing for structured P2P systems in an open environment without mutual trust is a huge challenge for all P2P systems. Firstly, the P2P overlay intends to provide an open, free-willing and administration-free overlay, which encourages all kinds of peers to cooperate together. Although it facilitates most normal nodes to make contribution to others, it also facilitates malicious ones to undermine the overlay. Malicious nodes can easily become a part of the overlay and commit sabotages without being punished or even known. Secondly, the Internet is a heterogeneous network made up of many networks of individual organizations/ISPs. Different organizations/ISPs have different network configurations and policies, such as IP address settings (dynamic or static,

fake or genuine), Network Address Translation (NAT), and firewall policies. Thirdly, various attacking methods from malicious nodes also increase the difficulty. Malicious nodes may simply drop or corrupt messages as they pass, or deliver to another malicious node instead of the legitimate replica root. Moreover, malicious ones may choose their node IDs by themselves, or even obtain a large number of legitimate node IDs by applying many times (Sybil attack) or by swapping IDs with complicities. All these reasons make P2P systems very vulnerable for attacking.

Much research [16, 17, 18] has been done on secure routing and object storage/retrieval. Since the routing is the basic operation for all the others, secure routing is essential for a secure overlay. It guarantees that the replicas are initially placed on legitimate replica roots, and that a lookup message reaches a replica if one exists. Similarly, secure routing can be used to build other secure services. However, due to the complexity of the P2P environment, many solutions are not satisfied, and some problems are still open, such as the Sybil attack [17] and the collusion of malicious nodes under dynamic IP environment.

As previous research [16] points out, the secure routing needs to solve three problems: securely assigning node IDs to peers, securely maintaining the routing tables, and securely forwarding messages. Among the three, to securely assign node ID is fundamental to the other two. To identify each principal (node) from the others is essential to securely assign node IDs. The IP addresses are currently used as identifiers of nodes. However, the IP address is not a good identifier under the P2P environment. More than 40% of nodes do not have true IP addresses or change their IP addresses from time to time [19]. Also, researches [16, 17] have even pointed out that Sybil attack and collusion among malicious nodes are unavoidable under a environment with dynamic IP assignment, even though the assignment of node IDs is delegated to a central, trusted authority (CA).

It seems impossible to identify each node from the others under the P2P environ-

ment, because of the administration-free P2P systems, varied network configuration of ISPs and all kinds of attacking methods of malicious nodes. However, a trust administration and authority does exist under such environment, and can be exploited to authenticate overlay peers. It is the network configuration and administration provided by each ISP or any organization. Although nodes may change their IDs or IPs, they can not change the network configuration imposed by network administrators. By exploiting authority of network configuration within our novel SNSA mechanism, a self-certificating identifier is assigned to each node, and the node can be easily checked by others. Based on this self-certificating mechanism, routing is guaranteed to be secure and malicious nodes can be tracked or punished.

# Chapter 2

# Network Structure and Related Techniques

In this chapter, the pros and cons of the Internet physical network structure and different network-aware techniques are discussed. At last, we propose a novel Scalable Network Structure Aware (SNSA) technique, which can provide sufficient information about physical network structure and locality of P2P peers.

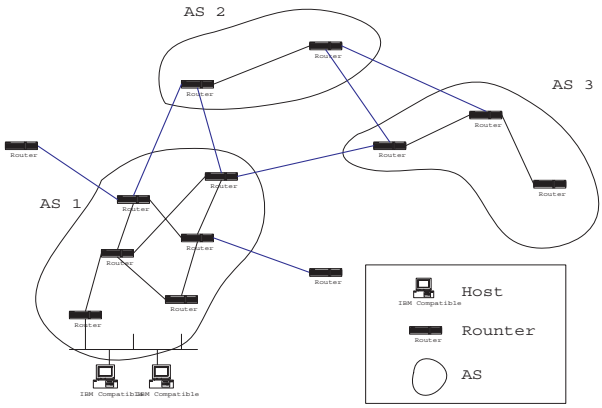


Figure 2.1: Autonomous Systems of the Internet. *Adapted from [1].*

## 2.1 The Structure of the Internet

It is well known that the Internet is made up of many Autonomous Systems (ASes). An Autonomous System (AS) is a collection of IP networks that is controlled by a single administration authority on behalf of an entity, such as a university, a business enterprise, or an Internet service provider (ISP). Normally, a common Interior Gateway Protocol (IGP), such as Routing Information Protocol (RIP) [20] and Open Shortest Path First (OSPF) [21], is used for routers to route packets between networks within the AS. Some border routers running Border Gateway Protocol (BGP) [22] connect the AS with its neighboring ones to form the Internet, just as figure 2.1 shows. Hosts within an AS are normally geographically close and connected by Local Area Network (LAN) or Metropolitan Area Network (MAN) techniques. Also, the connection of those ASes represents the power-law. That is to say the number of ASes, $f_d$, which have $d$ connections, is proportional to $d$ to the power of a constant $\theta$: $f_d \propto d^{\theta}$, where $\theta$ is around -2.2. In particular, some core ASes connect lots of normal ASes as well as other core ASes, while most ASes only have one or two connections.

The previous studies [23, 24] have shown that the structure of the Internet could be classified into three levels, *IP-level*, *router-level* and *AS-level*, based on different granularity. For IP-level, the granularity is too small to reflect the Internet structure. Also, it is almost impossible to obtain such a level of topology for the sheer number of IP addresses and a large number of machines leaving and joining at any given moment. The granularity of router-level topology can reflect Internet structure in detail. The BGP routing table is the most valuable resource to infer the Internet topology at this level. However, to acquire such information requires either the privileged access to BGP border routers or a lot of probing. Both of them are not a general way to applicant end users. As AS-level, the Internet has an even clearer topology with many known characteristics, such as the power-law. Also, many public services provide information such as the CIDR Report [25] and the IRR [26].

Although not as detailed as BGP routing tables, the information is publicly available and easily obtained. The drawback of the AS-level topology is that it is too coarse to provide detailed network structure inside, especially for large ASes.

Which of the above is the best to provide network structure information for P2P overlays? None of them seem perfect. The IP-level structure is too detailed and almost impossible. To my best knowledge, such level topology of the Internet has not appeared. Although router-level topology is very suitable to optimize large distributed applications, such as the well known Andrew file system (AFS) from Carnegie Mellon University, not only is it impossible for the whole Internet, but also it is overkill for most of the cases in P2P environment. Previous researches [27, 12] have stated that the distribution of peers in the real world is highly skewed. The nodes of the most popular P2P overlays are distributed in 4 to 5.5 thousand ASes, which is less than 1/3 of the total number of ASes in the Internet. Moreover, the node density is far from uniform. About 40% nodes were distributed within 10 top ASes, and normally node density of an AS is around 50 to 200. The AS-level topology provides a clear frame of the Internet and is easier for acquirement. However, it is too coarse for top ASes to provide further information of nodes.

## 2.2 Network Aware Techniques

There are many methods and techniques to acquire network-aware information, such as BGP routing tables [28, 29, 25, 26], "traceroute" tools, widely used landmark techniques [30, 31] and the network of physical springs [32]. In terms of provided information, those techniques can be classified into two categories: genuine network structure and relative network distance.

To provide genuine network structure information, BGP routing tables are the most powerful. It can reveal an accurate Internet structure for routers and ASes in detail.

| Network | Next Hop | Path |
|---|---|---|
| *8.0.0.0 | 245.35.236.4 | 0 2631 2 i |
| | 215.57.241.254 | 0 4548 6513 2 i |
| | 207.53.253.59 | 0 2738 3356 701 2 i |
| | 125.1.1.56 | 0 4444 386 2 i |
| | 173.28.240.90 | 0 2341 2 i |

Table 2.1: Example of BGP routing table entry

As mentioned earlier, the Internet is formed by thousands of ASes with border routers running BGP, which connect one AS with its physical neighbors. Two ASes are connected when there is a direct BGP link between them. By exchanging information with other BGP routers, each one forms a detailed routing table with AS paths to every existing AS. As a result, BGP routing tables can provide real Internet topology in detail, as table 2.1 shows. The advantages of BGP tables are obvious. However, they have their disadvantages. Firstly, BGP routing tables are not easily obtained. Secondly, because BGP tables include lots of routing related information, those tables are always too large and complex to be directly used. Fortunately, public services, such as the CIDR Report [25] and the WHOIS service from the Internet Routing Registry (IRR) [26], announce that information at AS-level. Much valuable information, such as the AS number, connectivity and IP address range, is listed in details. The CIDR Report even updates that information daily and provides thorough analyses.

Also, "traceroute" tools can provide router-level network topology. By sending out probe packets with step-increasing TTLs and receiving ICMP error messages from passing routers, those tools can expose the router-level path to the destination. However, this method not only suffers from high overhead and long latency, but is also infeasible for

many cases because ICMP packets are normally prohibited by network administrators in the firewalls due to security considerations.

By directly measuring RTTs between peers, landmark techniques and the Vivaldi have benefits for no dependency on supporting the underlying networks or accessing certain severs. They can provide network coordinates to reflect each node's relative network distance in the Internet. Different from network topology techniques, which provide actual structure information, they belong to network positioning ones, which always use RTTs between one node and other famous or pre-selected hosts, called landmarks, to calculate network coordinates. The Vivaldi also belongs to network positioning techniques, since it is also based on RTTs. However, it handles them by the spring network model, which has the ability to make the coordinates converge quickly and to predict relative distance between peers. While the RTTs can be affected by many unpredictable factors, such as changes of routing, link bandwidth and network traffic, those techniques are fully distributed and self-dependent, which are especially suitable for the P2P environment.

### 2.2.1 Limitations of RTTs based Techniques

Due to the advantages of RTTs-based techniques mentioned above, current P2P overlays always utilize relative distance among peers to optimize their overlay structures. Varied techniques of Proximity Neighbors Selection(PNS) have been proposed for Pastry, Tapestry and Chord. Although RTTs methods are suitable for P2P environment, lots of important network information of peers, such as available bandwidth and connectivity, can not be inferred from RTTs. For example, due to the instability of RTTs, current techniques tend to use the minimum-wise method. Although the minimum value may reflect the network distance between a pair of peers, its available bandwidth can not be estimated. It is easy to understand that a pair of peers may have very short RTT time while just across congesting routers. In addition, the peers with dial-up connections may have excellent

RTTs, but they are actually bottle necks. On the other hand, RTTs can not provide the information of connectivity between peers, since two peers may stay within two nearby ASes with firewall/Network Address Translation(NAT) traversal.

Actually, bandwidth and connectivity are crucial properties for P2P applications. It is well known that P2P applications always consume lots of bandwidth. Previous research [9, 12] has pointed out nearly one third of bandwidth of Internet back bones were consumed by all kinds of P2P applications, and they are still increasing. Saving bandwidth has become an important issue for P2P overlays. On the other hand, the connectivity of peers is also important. Due to wide deployment of firewalls and NATs, connections between peers behind different firewalls are very difficult. That is to say a peer may have lots of neighbors nearby (for short RTTs), but none of them is reachable.

With network structure information, properties of bandwidth and connectivity between peers can be estimated or inferred easily. By checking the path between two peers, their bandwidth can be estimated by the number and type of routers passing by. Normally, fewer routers and fewer border routers (normally congesting) means more bandwidth. For connectivity, nodes can be easily found out whether or not they are under the same firewall by their AS/ISP residences. In addition, the network structure information is more stable than RTTs based ones.

## 2.3 A Scalable Network Structure Aware Technique for P2P Overlays

The information provided by network structure is more stable and more valuable compared with RTTs methods. However, acquiring such information is not easy. In order to get router-level Internet structure, either BGP routing tables or "traceroute" tools are needed. Neither of those two methods is feasible in a P2P environment. The former needs

the privileged access to BGP border routers, and the latter involves a huge overhead of probing. Also, such detailed Internet structure is overkill for most of cases, as mentioned earlier. Thus, a particular method is needed for nodes of P2P overlays to easily acquire information of network structure.

Before determining the network-ware technique for a particular distributed application, we need to understand what information is the most important. As analyzed earlier, the essential network-aware information for P2P overlays should include *bandwidth*, *latency* and *connectivity* between peers. It seems that only router-level network structure information could satisfy these requirements. As mentioned earlier, such information is not only impossible, but also overkill. However, we find that *different levels* of network structure information can actually meet the requirements because of the highly skewed distribution of overlay peers under P2P environment. For the top ASes, which include thousands of peers, the router-level information is deserved. For midsized ASes, which host several hundreds peers, the brief structure information is sufficient. For small ASes, which have tens nodes or less, AS level information is considered enough.

Based on the observations, a novel method is proposed to provide sufficient network structure and accurate network locality of overlay peers under P2P environment. The basic idea of this method is to integrate various network-aware techniques based on different situations of overlay peers. The AS is considered to provide the *basic level* for network structure and locality of peers. This is because the AS provides an administration/connectivity border of peers within the Internet, and peers within the same intranet normally have more available bandwidth and shorter latency compared with the ones crossing the Internet. Also, acquiring such information is easy and simple. Based on public services, such as the CIDR Report and IRR, the AS of a peer can be directly found out by its IP address, and the overhead is trivial.

Since many ASes may include several hundreds of peers, the *middle level* informa-

tion with further network structure and locality within the AS is needed. The router-level information is obviously overkill; thus, an adapted landmark technique is utilized to explore network structure within an AS. Ratnasamy et al. in [31] have proposed a technique to provide each node with a landmark vector, which is an ordering of RTTs of a node to landmarks. By a *bin* algorithm, locality nearby nodes can be classed into the same bin. Although this technique shares the drawbacks with other RTT-based techniques as mentioned earlier, deploying it within an AS successfully circumvents those difficulties. By avoiding congesting border routers and changing routes, RTTs within the AS become much more stable and a good criterion to estimate the available bandwidth between peers. By their simulation in [31], 8 randomly selected landmarks, which can efficiently tell more than 1,000 nodes within a network, are considered enough to provide sufficient network locality of peers within a midsized AS. The suitable candidates of landmarks could be previously joined nodes or their default routers. Since all the traffic is happened within the AS, the overhead of this method is low for both landmarks and peers.

For the top ASes, which have a very high density of nodes, even landmark vectors may not be enough to distinguish their network locality. As a result, the router-level network structure, *high level*, is needed. Instead of using "traceroute" tool, a more efficient technique, IP Record Route [33], is utilized. Instead of sending out many packets to discover routers along the path, the IP Recode Route protocol can ascertain up to 9 routers by one packet. Since all peers are within the same AS, recording 9 routers is sufficient for their paths. This technique may only be performed within an AS, since it is prohibited by most of the network administrators in the firewall. The overhead of this method is also not high. Firstly, the probing method is highly efficient. One packet with the IP Record Route protocol can detect all routers along the path. Secondly, the information can be highly shared. Due to high node density, many peers may be within the same LAN; the network structure information can be shared with others. Also, based on the known router-level

topology information, new peers can easily find and add their routers to known router topology only by detecting nearby ones.

The three-level network structure hierarchy above forms a scalable abstract for a highly skewed distribution of peers under a P2P environment. Not only does it provide sufficient and accurate network structure and locality information of peers, but also the overhead of this technique is low. The network-aware technique used for each level is almost the exact one to satisfy the requirement. Also, all techniques are fully distributed with little involvement of centralized services, which is especially important for the P2P environment. This technique will be mentioned as *SNSA technique* for the short of Scalable Network Structure Aware technique in the following chapters.

# Chapter 3

# Overlay Building

In this chapter, we will discuss in detail how to build a P2P overlay based on network structure of the Internet and network locality of overlay peers.

## 3.1 Network Locality under P2P Environment

It is well known that exploiting network locality is crucial to improving the efficiency of a distributed system. Traditionally, the *network locality* of nodes within a distributed application is defined as latencies between nodes. However, this definition needs to be extended for P2P applications. Nodes for traditional distributed applications, such as NFS and AFS, are normally within the same sub-network or Autonomous System, and based on administrator-maintained servers. However, peers for P2P applications are globally distributed and based on end-user computers without privilege. As a result, the connectivity between peers becomes a consideration because of wide deployment of firewall/NAT. Moreover, while nodes for traditional distributed applications are normally connected by LAN techniques and generate traffic within the intra-network, peers for P2P applications utilize all kinds of network connections (such as dial-up, cable modern/DSL and fast Ethernet)

and communicate with peers around the world. Not only does the difference of bandwidth between peers need to been taken in account, but also the impact of the network traffic to the whole Internet needs to be considered when building a P2P overlay.

Thus, we consider that, in addition to the latency, the *network locality* for P2P environment should take both *bandwidth* and *connectivity* of peers into account. As analysis in previous chapters stated, network locality information can only be acquired based on network structure information. Thus, we argue that an efficient P2P overlay should be based on network structure information, as well as network locality of peers.

## 3.2 Notion of the Overlay

In contrast to the current top-down designs of P2P overlays, we propose to use bottom-up method. The P2P overlay is built based on the network structure of the Internet and network locality of overlay peers. Compared with current designs, building an overlay to reflect network structure and locality of peers has at least two advantages. Firstly, by eliminating the inconsistencies between the overlay structure of systems and physical network characteristics of overlay peers, the efficiency of P2P systems is highly improved. Due to the theoretical approach, physical network characteristics of P2P overlay are ignored to a great extent. Although the theoretical approach facilitates the implementation of overlay functions, such as the elegant routing in structured systems, it trades off the efficiency of the whole system. Because of the structure inconsistency, logical neighbors tend to be physically far from each other. Since logical neighboring peers obviously have much more communication compared with others, common operations, such as routing, become costly. The latest P2P designs try to take network-aware information into account; however, the essential difference between regular system structure and the unique distribution of overlay peers makes this effort difficult.

Secondly, exploiting network locality of peers can provide sufficient bandwidth and full connectivity to make systems more powerful and dependable. The system churn is a large challenge for P2P systems. Although both structured and unstructured overlays can tolerate it to some extent, research [9, 12, 10] has pointed out that the churn in a practical environment is very intense. Also, lots of Internet traffic is generated by operations to maintain routing and neighboring tables for structured and unstructured systems. By fully exploiting network locality, not only will most of maintenance traffic take place within the intra-network instead of crossing the Internet, but also full connectivity between peers makes it possible to deploy more powerful protocols to improve dependability and security of systems. Many techniques, which are normally used in LAN, such as broadcast and erasure coding, can be deployed because of sufficient bandwidth and connectivity within the intra-network.

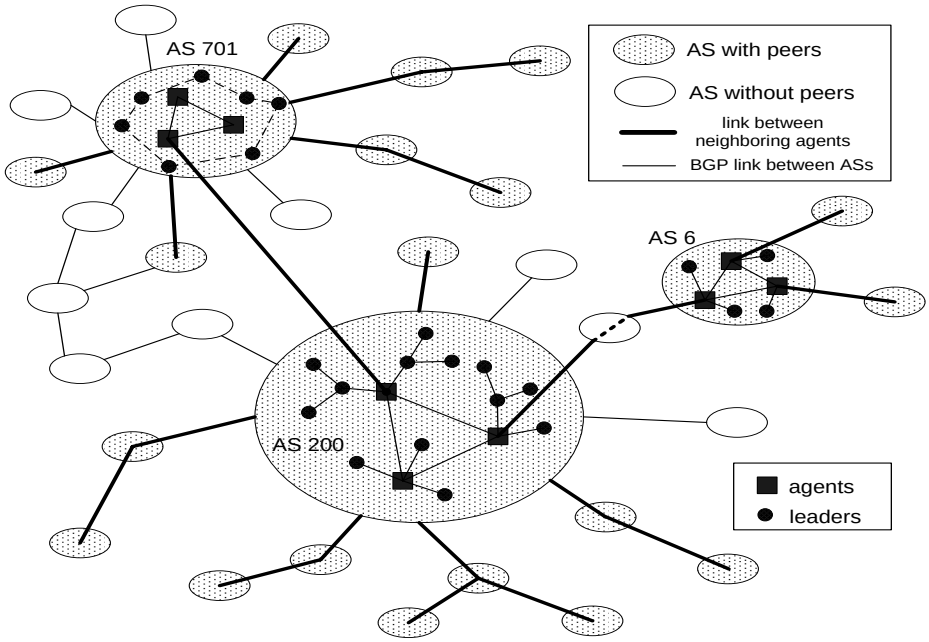


Figure 3.1: Building the system overlay closely matching Internet topology

## 3.3 To Build the Overlay

In order to build a P2P overlay direct which directly reflects network structure and locality of peers, we organize our overlay following the Internet structure. Actually, the Internet itself is a nested P2P-like system. It is well known that the Internet is a collection of many autonomous networks (ASes). Those networks link with some others by WAN techniques and finally form a globally connected network, the Internet. At this point, the Internet is a P2P system for AS. Also, within each network (AS), there are many sub-networks, which are made up of many hosts connected by LAN/MAN techniques. That is to say, an AS is made up of sub-network peers.

Similar to the structure of the Internet, all peers in our overlay will firstly be divided into *groups* by their AS locus, as figure 3.1 shows. This information can be easily found out by IP address of a peer, which is also the basic level of information provided by the SNSA technique. Since the distribution of peers are highly skewed, partitioning peers only by their AS locus is too coarse in many cases. Thus, we further divide nodes within a group into *teams*, which are like the sub-networks within an AS. Based on the node density of the AS, different techniques are utilized. By involving the middle level SNSA technique mentioned earlier, the landmarks technique will be exploited for most cases, which normally have hundreds of peers. By randomly selecting 8 landmarks, each node can form its landmark vector by measuring RTTs from landmarks. By calculating the relative distance from peers' landmark vectors, physically nearby nodes will be grouped into one team. For the AS with an even higher host density, the high level SNSA technique is exploited to provide the router-level network structure. The nodes under the same router will be clustered as a team, which actually follows the physical structure of sub-networks.

Some peers with high bandwidth and availability will be selected as agents to connect with other agents and manage teams. Also, a leader will be elected to serve team-

mates and connect with agents. In fact, this group (agent)-team (leader) structure is also the pattern of the physical network. For example, the Internet (P2P overlay) is made up of ASes (groups), and each AS consists of many sub-networks (teams). The AS has some border routers (agents), and each sub-network has its default router (leader). Normally, agents will keep two tables, one for communicating with other groups and the other for in-side teams. The leader will keep a list of its teammates, and connect with a nearby agent, as figure 3.1 shows. Only agents and leaders are involved in core operations, such as routing; other peers provide auxiliary works. The details of data structures and procedures are subject to change for different system designs, which will be discussed in detail in next chapter.

Basically, the nodes within one AS will be clustered into one group for the consideration of network locality. If an AS is too large in node number, dimensionality or both, the peers will be divided into several groups following the network locality. Also, for the nodes in very small ASes, instead of forming their own group, they will join a physically nearby group as teams.

It seems that our overlay introduces some server-like roles, such as agents and leaders. Actually, we exploit widely existing server-like nodes within in P2P systems instead of introducing them. Although P2P protocols try to grant nodes equal responsibility, previous research [9, 12] has revealed that behaviors and resources of peers vary extremely, and server-like nodes do exist. While most nodes have very short uptime, there are about 10% nodes with more than 90% uptime. Also, those 10% hosts also contribute about 90% of the total traffic. Our simulations show server-like peers under a synthetical environment. Our simulations of Chord have also showed there were 5% server-like nodes, which had much longer uptime and higher reference number, even under the random distribution of node ID and life time.

While leaders and agents consume more resources than normal ones, our exper-

iments and analysis, which is discussed later, show that not only is the efficiency of the whole system highly improved, but also the extra work for leaders and agents is not a burden for modern computers. By excluding transitory and weak peers out of core operations, system efficiency and availability are highly improved. Not only is each step during routing much quicker and more reliable, but also the total number of steps decreases. Also, the maintenance overhead of the overlay is highly reduced. Moreover, by carefully adjusting overlay parameters, such as group size $G_{sz}$ and team size $T_{sz}$, the extra overhead of agents and leaders can be controlled, which is detailed discussed in chapter 4.

## 3.4 Membership Management

Under P2P environment, not only is each node's leaving and failure unpredictable, but also the whole overlay is highly dynamic. Since each node can arbitrarily join and leave the system without any restriction, the frequency of nodes joining and leaving is very high. Although agents and leaders are considered to be more dependable than normal nodes, they are not well-maintained servers. They can fail or arbitrarily leave the system. The leaving of agents and leaders will not impact the overlay much. A nearby node can quickly replace the leaving agent or leader, as the one sends out its leaving message. Also, the overhead is trivial, which just involves meta-data copying within the intra-network. However, the situation of crash is different. Although crash of the node is not considered a common case, it should be quickly detected and recovered. A modified *Ring* protocol [34] is used in our overlay to monitor agents and leaders. All leaders will form a ring as in AS 701 shown in figure 3.1. Every second each one will send a keep-alive message to its successor and predecessor. Although this protocol is simple, it is efficient enough to detect one or more than one failures. For example, within a ring of four leaders, A, B, C and D, assuming leader C is dead, leader B and D will know within one second and report to agents. Even

if leader B and C are failed at the same time, leader A and D will report to agents. The same method is also used between agents. For management of teammates, the leader can monitor their changes during their normal communications. However, in order to accurately distinguish each peer's behavior for candidates of leaders and agents, each peer is designed to send either a query or a keep live message to its leader periodically, which is discussed in the next chapter.

## 3.5 Node Joining and Leaving

When a node joins the system for the first time, its AS number can be determined by its IP address through our SNSA technique. By any node within the overlay, the joining request can be forwarded to one agent within the AS of the new node. Either by the middle level or high level information of the SNSA technique, the node's network locality can be determined. Then, the node is assigned to an appropriate team. The overhead of the joining is minimal, since only the leader updates some book-keeping information.

When a team is too populous, it will split into two teams based on network locality. If the joining node is the first one in that AS, the requirement will be forwarded to an agent in the group nearby. The node will initially become a normal node within that group, called *mother group*, instead of forming a new one. When the number of nodes is sufficient for a group, agents will be selected and an individual new group is born. The mother group will announce the new one to all others.

The leaving or failure of normal nodes is automatically tolerated by the team. If nodes keep leaving a team, the team will disappear. The remainder will join a nearby team. The situation will be similar for a group's disappearing. If a leader or agent is leaving, a new one will be selected, and the meta-data will be quickly rebuilt or copied.

## 3.6 Summary

The above is a general P2P overlay based on the physical network and locality of overlay peers. By cooperating with addition data structures and operations, it can be constructed to distributed systems with different purposes. In the following chapter, we will discuss how to enhance the overlay to support structured and unstructured P2P systems.

# Chapter 4

# Designs of Structured and Unstructured P2P Systems

Based on the overlay mentioned in last chapter, different data structures and operations can be loaded to support different applications. In this chapter, we will discuss how to exploit the overlay to implement highly efficient structured and unstructured P2P systems.

## 4.1 Structured P2P System Design

The essential feature of structured P2P systems is to map each object to a certain place. The keys to implement this feature are two parts: *ID mapping mechanism* and *ID-based routing*.

For the ID mapping mechanism, by assigning each object and node a unique ID, current designs map each object to a node whose ID has a unique relationship with the object ID. For example, each object is mapped to the node with the closest ID in Pastry, and to the closest predecessor in Chord. For our system, a similar ID mechanism is exploited.

Each object in our system has a 128-bit ID, which can be generated by a basic hash function such as SHA-1 [35] like Chord and Pastry. Different from current designs, only groups and teams have IDs. Following our group-team structure, two-level ID and mapping mechanism are built. For group level, each group is assigned a 32bits ID as its group ID. Since valid Internet AS number is from 1 to 64511 and there are only about 17,000 active ones currently, we believe 32bits ID is enough for groups. For team level, each team also has a 32bits ID. Given a 128bits object ID, the first 32 bits decides which group it belongs to, and the second 32 bits determine the exact team it will be located within the group. In other words, each group will charge the objects with the first 32bits falling in the range between its group ID and the next one, which is similar with the mapping of Chord. Also, the mapping is similar for teams within the group.

For the ID-based routing, by keeping a routing table with $O(LogN)$ entries in each node, current designs (DHTs) guarantee to deliver a message to the destination within $O(LogN)$ hops with N the total number of nodes in the system. However, due to the varied resources and behaviors of nods, DHT designs face many difficulties in routing efficiency and high overhead under system churn. Also, some research [36] points out that the DHT may only get benefits for the overlay with more than $10^7$ nodes. Actually, the largest deployed P2P system has no more than two millions nodes [12]. As a result, we plan to directly utilize the Internet IP routing, instead of building a new overlay routing. Following our two-level mapping mechanism, delivering a message will be made up of two straightforward steps. The message will be sent to its target group, and then be forwarded to its destination team. Further details will be discussed in the next section.

The team is the basic unit to store objects. At least two copies are kept within a team for improved dependability. One copy is kept in the leader to respond to queries quickly. The other is striped into blocks by n erasure code technique and stored among teammates. Major nodes with asymmetric network connections, such as DSL and cable

modem, make this scheme especially efficient. Previous research [37, 14] has pointed out that erasure code technique can not only significantly improve availability and reliability of objects under P2P environment, but also reduce bandwidth consumption for updating objects. The only drawback of erasure code within DHTs is the read latency [14]. However, our overlay successfully solves this problem. Since most of the nodes are connected with asymmetric network connections, reading from nearby nodes is evidently faster than reading from one. This is also useful to quickly recover datum for new leaders or agents when the old one is leaving.

### 4.1.1 Routing

Instead of involving many transitory and weak nodes in routing, only agents and leaders will be exploited. Three agents per group are considered enough to provide sufficient service without loading their hosts. Detailed analysis will be given in later section. Also, one leader and a backup will be selected in each team. The leader keeps a link with the nearest agent. Other nodes periodically report to their leader. Their behaviors are recorded by the leader for team health and candidates of agents and leaders in the future.

The agent is the key role for overlay routing. As mentioned earlier, since a two-level mapping/routing mechanism is used in our overlay, two tables are kept in agents for routing. One records every group's ID and their agents' information, called the *routing table*. This table is maintained in each agent, and cached by leaders to route messages for normal nodes. The other one is only kept in agents. It records every team's ID and leader's information within the group, called the *delivering table*. The routing procedure is made up of three steps and involves two hops. First, when a node wants to lookup an object, it simply drops a message to its leader. Second, by checking the first 32 bits of object ID in the routing table, the destination group can be found. Then the leader directly forwards the message to one agent within the destination group. Third, when the agent receives the

message, it uses the second 32bits to find the response team in the delivering table, and send the message to the responsible leader. Compared with current designs, the efficiency of routing is highly improved. For the number of hops, while the average hops to route a message in current structured systems is around 6, ours is two. Actually, one more hop means one more overhead of both network communication and process. For the routing latency, ours is near the ideal one, while the result of current designs is far from the ideal one. This is discussed in chapter 7 in detail.

### 4.1.2 Maintenance

Although agents and leaders are considered to have higher availability than normal nodes, they are not well-maintained servers. Their failings and leavings are monitored by the modified ring protocol [34], which is mentioned in the last chapter.

For the membership management of normal nodes, the leader can monitor their changes through query messages. However, in order to accurately distinguish each node's behavior for candidates of leaders and agents, each node is designed to send either a query or a keep live message to its leader every 30 seconds. Normally, the node with the longest session time will be become a backup of its leader. When an agent crashes, one leader will be upgraded to the agent. Since the leader also keeps the routing table, only the delivering table will be copied. When a leader crashes, the backup one will replace it. A copy of stored objects can be quickly rebuilt from teammates by the erasure code technique. Also, the latency of each teammate will be recorded by the leader to monitor the changes of physical network structure.

Since the routing table is the key for overlay routing, keeping it up to date is critically important for the correctness of overlay routing. Also the overhead of this maintenance should be low; otherwise, both performance and scalability of the overlay will be affected. Thus, our overlay is designed to integrate such works into the common operation,

*lookup*. When the leaders send out messages, latest information of changed agents will be appended to outgoing messages, and reach the agent in the target group. The agent then filters out those updates, and notifies the other two agents of the keep-alive messages by the ring protocol. After that, those updates are further piggybacked with query messages to leaders. In order to save bandwidth, the agent keeps an updating table for leaders, which records updated information, and only the useful update information will be forwarded. When routing tables within leaders have been updated, the information can be further sent out to more groups. This *gossip-style* piggyback mechanism is efficient and robust. It is well known that with high probability, all groups will be informed within $O(LogN)$ steps in which $N$ is the number of groups. Our experiments also prove it. Even under a highly dynamic environment, our overlay keeps a very high success rate, which will be shown in Chapter 7.

### 4.1.3 The Sizes of Groups and Teams

Although to group nodes based on the AS provides a clear overlay structure, previous research [12] has shown that the distribution of node density is highly skewed. The number of nodes within an AS, $G_{size}$, varies from tens of thousands to several. Simply organizing nodes within one AS to a group may involve serious load balance problems. For consideration of fairness, we would like to keep $G_{size}$ for each group almost the same. Thus, instead of forming an independent group, an AS with a small number of nodes will join a physical nearby group. Also, the AS with a large number of nodes will be divided into several groups.

Also, it is important to keep extra works for agents small; otherwise, users will avoid becoming agents. Since the network bandwidth is the most valuable resource for nodes, we restrict the extra bandwidth used by agents not to exceed 1% and 5% for download and upload, respectively. We assume the request rate of each node is $\gamma$ and the number

of bytes for each message is $M$. By assuming three agents per group, the bandwidth for an agent to forward messages is $\frac{G_{size} \cdot \gamma \cdot M}{3}$ for both up and down directions. Considering a typical DSL or cable modem connection with 3Mb downlink and 384Kb uplink, and assuming that $\gamma$ is 1 request per node per second and $M$ is 20 bytes, the $G_{size}$ should be the MIN(563, 360), in which 563 is the limit of download and 360 is the one of upload. Excluding forwarding messages, the ring protocol and piggyback technique also consume the upload bandwidth. Thus, a suitable $G_{size}$ should be around 300 for the limitation of upload. Actually, this is a modest estimation. Because the upload traffic of agents is only happening within the intra-net and the bandwidth limitation is controlled for the Internet traffic by ISPs on border routers, the actual upload bandwidth for agent is much higher than the limitation value for the intra-net.

The number of nodes within a team, $T_{size}$, is also a very important parameter for the overhead of leaders, dependability and efficiency of teams. Compared with the agent, the consumed bandwidth of the leader is relatively small. Thus, the bandwidth is not a major consideration for leaders. Although the availability of the erasure code technique is very high, it is seriously compromised by the frequent joining and leaving of nodes. Previous research [9] has shown that the average uptime of nodes is about 30% to 40%. In order to exploit asymmetric connections to quickly retrieve datum among teammates, the leader is considered to connect with at least 8 nodes at any time. Thus, the suitable $T_{size}$ should be around 20 to 30.

### 4.1.4 System Overhead Analysis

For real world systems, nodes are distributed into 4 to 5.5 thousand ASes and the number of nodes is from 200,000 to one million. Assuming $G_{sz}$ is 200, the number of the group is about 5,000. Thus, the size of the routing table with all groups and their agents' information is about 60KB, and the size of delivering table with 15 teams is about 0.2kB.

| System Functions | Agents (Bps) | | Leaders (Bps) | | Nodes (Bps) | |
|---|---|---|---|---|---|---|
| | Downlink | Uplink | Downlink | Uplink | Downlink | Uplink |
| Lookups | 1333 | 1333 | 400 | 400 | 20 | 20 |
| Member | 20 | 20 | 27 | 20 | 0 | 0.33 |
| Piggyback | 844 | 264 | 44 | 240 | 0 | 0 |
| Total | 2197 (0.58%*) | 1617 (3.36%*) | 471 (0.13%*) | 660 (1.4%*) | 10 | 10.33 |

Table 4.1: Bandwidth consumed by different roles and operations. * is the ratio to DSL or calble modem connection with 3Mb downlink and 384Kb uplink

They are obviously not a burden for modern computers.

Previous research [10, 38] has pointed out that nodes' joining and leaving under the P2P environment can be modeled by a Poisson process. Thus, an event rate $\lambda$ corresponds to a median inter-event period of $\ln 2\lambda$. Therefore a churn rate of $\lambda$ that corresponds to a median node session time of $N$ nodes within a network is the following:

$$t_{med} = N \ln 2/\lambda \tag{4.1}$$

Considering an overlay similar with a real world one, which has $10^6$ nodes distributed in 5,000 groups with 3 agents per group and the median session time of 15 minutes per agent, the churn rate of the 15,000 agents is 11 per second by Formula 4.1. Thus, the bandwidth for one agent to deliver the information to another agent is 44Bps, and to deliver to the 5 nearest leaders is 220Bps. By assuming an average of 20 nodes per team, 200 nodes a group, an average query rate of 1 per node/second, piggybacking three agents each time and 20 bytes per message, Table 4.1 shows the break down of the consumed bandwidth of all roles and functions of the system. As the table shows, our system can easily scale to a system with more than a million of peers.

## 4.2 Unstructured P2P System Design

The unstructured P2P systems can be further classed into three styles. One is the pure P2P one, in which peers act as clients and servers equally, such as Gnutella. Another is the hybrid one, which has a central server to keep information of peers and responds to requests, such as Napster. The third is a mixture of above two, such as KaZaA and Gnuetella2. The *supernode* technique is widely used in the mixed-style P2P systems. Those supernodes highly improve both scalability and query performance of unstructured P2P systems. Currently, KaZaA is the largest deployed P2P system.

For the unstructured P2P system based on our overlay, agents and leaders will naturally act as supernodes. The leader will index shared files of teammates, and report to the agent. In order to prevent the blind flooding in Gnutella, a technique called Query Hash Table (QHT) is introduced in Gnutella2. A QHT is a table of $2^N$ bits, where each bit represents a unique word-hash value. When a searchable plain-text word is contained in a node's content, the related bit is marked. Actually, the QHT provides enough information to know with certainty that a particular node (and possibly its descendants) will not be able to provide any matching objects for a given query. Normally the $N$ is 20, which has more than one million possible word hash values. The uncompressed QHT is 128KB in size. Our system will adopt this technique to facilitate the query operation. Each leader will periodically collect QHTs from teammates. After combining those tables into a team-level QHT, the leader will send it to the nearest agent. The agent will keep the QHT of each team, and also form a group-level QHT by combining them together.

Due to the power law of the Internet, some ASes are hub-like. They always connect hundreds of ASes. By the Internet AS-level topology provided by the CIDR, we also find that those hub-like ASes are highly connected with each other. The average AS path length among the top 30 hub ASes is 1.47. Actually, the Internet topology presents a small world

effect. Every AS can reach at least one of the hub ASes within the AS path length of four, and the diameter of the Internet is less than 10 at AS-level. This characteristic can be further exploited to organize groups in our overlay. That is to say, the groups around a certain hub AS can be organized into an *area*. The group within the central position of the area will become the hub group. The agents within the area will send their group-level QHTs to the hub group. After joining those QHTs to an area-level one, the agent of the hub group will share it with other hub groups. The technique to share one's index with neighbors is called *host cache*, which is widely used by unstructured P2P systems to improve the efficiency of the search process.

Instead of blind flooding, the search procedure follows the *expanding ring* model. For example, when a node wants to do a search, the request will firstly expand to the team by querying the leader, and then enlarge to the group by the agent. If the result is still unsatisfied, the request will be forwarded to the hub agent and swell to the area. After that, the request can be further sent to hub agents of nearby areas and finally spread through the whole system. During this procedure, the QHTs are utilized to quickly locate related nodes and reduce unnecessary forwarding. Moreover, the host cache will be exploited to further improve the efficiency of the search. For example, each leader can cache other teams' QHTs within the same group, and the agents can get other groups within the area. As the result, when a request reaches a leader, it essentially expands to the whole group, and to the area as it comes to an agent.

Of course, the above is a brief design for unstructured systems. The latest techniques mentioned in [39], such as topology adaptation, flow control, and random walk search, can be adopted and can take benefits from our overlay and the SNSA technique.

Most of the work within this chapter was published in [40]. Part of it was published in [41, 42]

# Chapter 5

# Secure Routing for the Structured Design

In order to fully utilize the potential of P2P systems, structured P2P overlays must be able to tolerate the open Internet environment where mutually distrusting parties with conflicting interests are allowed to join. Structured overlays must be robust to a variety of security attacks, including the case where a fraction of the participating nodes act maliciously. Such nodes may misroute, corrupt, or drop passing messages and information. Additionally, they may attempt to assume the identity of other nodes and corrupt or delete objects they are supposed to store on behalf of the system.

However, due to the notion of administration-free for P2P overlays, varied network configurations of ISPs and all kinds of attacking methods of malicious nodes, it is a huge challenge to guarantee security under a P2P environment. Many research [16, 17, 18, 43] has been done on object storage/retrieval and especially secure routing, since it is essential for all the other operations. It guarantees that the replicas are initially placed on legitimate replica roots and that a lookup message reaches a replica if one exists. Moreover, secure routing can be used to build other secure services, such as maintaining file metadata and

user quotas in a distributed storage utility.

Two levels of communication are involved with the routing operation of a P2P overlay. One is at *network-level*, where nodes communicate directly through an underlying physical network, such as TCP/IP. The other is at *overlay-level*, where messages are routed through the overlay using P2P protocols, such as Chord and Pastry. Many techniques have been developed to prevent adversaries from observing or modifying network-level communications. In this dissertation, we only address security issues at overlay-level.

As Castro, et al mentioned [16], the secure routing is needed to solve three problems: securely assigning node IDs to peers, securely maintaining the routing tables, and securely forwarding messages. Although many solutions have been proposed, they have their difficulties or limitations. For securely assigning node IDs to peers, in order to prevent attackers from choosing node IDs, the current solution is to delegate this problem to a central, trusted authority (CAs). They sign a certificate for each node ID, which binds a random node ID to the public key that speaks for its principle and an IP address. However, this solution is not practical because more than 40% of peers either do not have true IP address or change their IP addresses from time to time [19]. In order to prevent attackers from easily obtaining a large number of legitimate IDs (Sybil attack), the current solution is to charge money for each certificate. However, this solution may thwart users' enthusiasm. Moreover, as previous research [17] has pointed out, in P2P systems where IP addresses are allowed to change dynamically, node ID swapping and Sybil attacks may be unavoidable. For securely maintaining the routing tables, the current solution is to impose strong constraints on node IDs in routing table entries instead of proximity neighbor selection (PNS). This solution reduces the possibility of malicious nodes within the routing table. Not only does it impact system performance, but it is also a conservative method to prevent thing from going worse. Otherwise, a small fraction of malicious nodes may control a large number of peers under PNS. For securely forwarding messages, current solutions are to detect

faults and use diverse routes. The detection of faults is based on the assumption that the density of node IDs per unit in the ID space is uniform. However, it may not hold in the real world. Also, the diverse routes involve two or three times the overhead for each message routing.

In summary, due to the complexity of the P2P environment, current solutions are either impractical or not strong enough or involving too much overhead. Moreover, the issues about Sybil and node ID swapping attacks under a dynamic IP environment are still open.

## 5.1 A Better Identifier

Identifying each principal (node) from the others is an essential issue for the security of all systems. However, this is a really challenge under a P2P environment. Current P2P overlays normally identify nodes by their IP addresses. It is obviously not a good identifier. Due to varied network configurations from ISPs and connection techniques from users, more than 40% peers do not have a true IP address or change their IP address from time to time.

Based on the observation that each peer must subject to the network configuration and administration imposed by ISPs, we propose to identify each node by its physical network characteristic, called *net-print*, instead of the mutable IP address. The net-print is a set of information of the node's physical network characteristic, such as RTTs to some hosts, its external IP and setting of its default router, which are imposed by their ISPs. Those characteristics can be naturally detected by our SNSA technique mentioned in the previous chapter. Two reasons make the detection accurate and efficient. Since the nodes are organized by network locality, they are normally within the same AS and are physically close to each other. First, the detection of RTTs will be more accurate and more efficient

compared with global landmark detections. Second, many powerful network protocols, such as ICMP and ARP, which are normally prohibited by network administrator in the firewall for security considerations, can be exploited to provide more detailed information of physical network configurations, such as a node's Media Access Control (MAC) address, intra-network IP and its default router settings. That valuable information can locate a peer to the sub-network level or even switcher level.

In order to cooperate with our SNSA technique, the net-print of a node includes AS number/IP range, landmark vector within the AS, MAC and the default router IP of the node. The procedure to collect the net-print of a node is as follows. As a new one wants to join the overlay, it must have an introducer. By contacting an introducer outside the AS, the AS number/IP range of the new one can be found out from its external IP address. Then the joining message is forwarded to one agent within the AS. Before the new node joins a team, the agent will require it to report its landmark vector by measuring the RTTs to several routers. At the same time, the agent will randomly select some nodes, called detectors, within the sub-network of those routers to measure the RTT to the new node. Instead of directly using the vector claimed by the node, a net-print vector is formed by the RTTs from detectors. By comparing the two vectors, the cheating of a node can be easily detected. For the AS with very high node density, more accurate locating information may be needed. Since all those nodes are within the same intra-net, a node's default router IP address can be found by an ICMP packet with the routing record option from any other node. Because the IP addresses and policies of routers are charged by network administrators, not only peers can not change them, but also any violation of those policies make peers disconnected from the network. By the Address Resolution Protocol (ARP), the MAC address of a peer can be discovered. With the support of Simple Network Management Protocol (SNMP), we can even locate the switch of each peer.

Actually, the net-print itself is *self-certifying*. That is to say it can be directly

verified by others. For example, one node $M$ wants to pretend to be another node $A$, and even know the net-print information of $A$; however, other nodes can easily verify $M$ by comparing the claimed net-print and directly measured one. Moreover, it can be used to thwart the most difficult issues about P2P security, Sybil attack and node ID swapping attack (or collusion of malicious nodes, in another word). With the help of the net-print, the node ID swapping attack across ASes is impossible, since it can be easily detected from the mismatch between the external IP address and the AS number. Also, the collusion within the same AS is very difficult, since landmark vectors and a default router IP can detect it. Although, the malicious nodes under the same sub-network may still collude with each other, the net-print efficiently locks them within a small network scope and restricts the number of colluded ones. For the Sybil attack, though the malicious node may pretend to be different nodes under a sub-network, not only the number of faked peers is restricted, but they can also be identified by challenging those nodes to solve a unique computational puzzle concurrently.

## 5.2 Secure Routing

Based on the better identifier, net-print, we will discuss secure routing issues of our structured P2P design in detail. The overlay network runs on a set of $N$ nodes that form an overlay using the protocol described in the previous section. We assume a bound $f$ $(0 < f < 0.5)$ on a fraction of nodes for every role, such as agents and leaders, that may be faulty. In the following paragraphs, we will discuss the three key issues to implement secure routing: securely assigning node IDs to nodes, securely maintaining the routing tables, and securely forwarding messages.

First, securely assigning node IDs to nodes is discussed. Due to the self-certifying characteristic of the net-print, collusion and Sybil attack are almost impossible in our sys-

tem. Also, the effect of regular malicious nodes is very little, since they only store parts of objects and their failures are tolerated by the erasure code. Although previous research [16] has pointed out that giving the P2P overlay a public key infrastructure is important for system security, to give each node a certificate is obviously not necessary in our system. Instead, only the agents will be granted a certificate by a trusted certification authority (CA), which binds a public key to its net-print instead of its IP address. Besides the normal properties of net-print mentioned earlier, the net-print for agents also includes its current interior IP, exterior IP and port for routing service. Even though the agent is under an ISP using DHCP or NAT, that information can identify it during its service time. Those public keys will be spread to all agents and updated with the changes of agents. The leader's certificate can be granted by any agent within that group. It includes interior, exterior IP and team ID.

Second, we discuss how to securely maintain the routing table. As mentioned earlier, the maintenance of routing tables in agents is based on the piggybacked information within the messages. In order to make this procedure secure, we exploit the redundancy and electric signature techniques together. The detailed procedure is as follows. When an agent filters out the updated information from an incoming message, it will firstly check the certificate of the message and then share the information with the other two agents by the ring protocol. Due to the gossip-style mechanism of updating agents, the update information about one agent can be received from different resources. Thus, those updates will be inconsistent if a node tries to forge it. If no conflict is detected, this update will be accepted by all agents, and delivered to leaders. Also, this updated information will be further piggybacked in outgoing messages by leaders.

Third, based on the above discussion, the procedure of forwarding messages securely in our system is straightforward. When a node requires its leader to route a message, it will first check the leader's certificate and then send the message. Then, the leader will

randomly choose one agent within the destination group to drop the message with its certificate. As the destination agent receives the message, it will check the IP and port number with the certificate to make sure it comes from the expected leader. Then it signs the message and forwards it to the responsible destination leader. Finally, the destination leader will send back the result with its certificate. Normally, the node can safely use the information. If the result is found out to be incorrect, the node can provide the message to an agent. By tracking the message, the faulty node can be found out, since the leader for the required result may itself be faulty. The replica will be reached by the key generated by the hash function initialized with another seed, which is similar to the technique used in CFS [44]. Instead of just tolerating faulty nodes like current designs, those nodes will be exposed. Any node can report a suspect faulty leader or agent to its agent with the evidence of the result of redundant routing or the tracking back message. If that node is continually impeached by different nodes, the agent will consider it fault and keep it in a *black list*.

## 5.3 Some Discussions

Although malicious nodes do not have many chances to interfere with our system under the security routing above, we still need to prevent the accumulation of malicious agents and leaders. Since many malicious nodes may have more powerful CPUs and longer session times, they may be good candidates for management positions. The solution to this is to limit the term of every leader and agent. A maximum period $P_{max}$ will be imposed for every lead and agent. When the maximum period is reached, a new one will be elected to replace it. Since a leader's certificate may just be issued by an expiring agent, the public key for each agent will be kept for another $P_{max}$ as the grace period. After that, the public key will be deleted. A previous agent may be selected to be an agent again and a new

certificate will be issued. The $P_{max}$ should not be too long; 24 hours could be a modest value.

In addition to limiting the maximum term of leaders and agents, our overlay can even eject the misbehaving nodes. As mentioned earlier, agents will keep a black list for malicious agents and leaders. If a malicious agent or leader is on the black lists of two or more agents of one group, they may accuse that node jointly. They will send an appeal with their signatures to an agent in that group. Then, the term of that node will be reduced. The more appeals received, the more quickly the node will be ejected. Finally, a new leader or agent will be selected to replace the malicious one, and its certificate will be revoked. As the result, that node will be isolated by the others. All nodes will stop sending messages to that node and ignoring messages from it.

Most of the work within this chapter was published in [45, 46].

# Chapter 6

# A Common Overlay Network Under-layer

Although our network-based overlay network can be developed into highly efficient and secure structured and unstructured P2P systems, it is no a panacea. More P2P systems and other overlay networks will keep emerging. New requirements and features will be proposed. However, among changes of overlay networks, routing is their essential issue. Optimizing overlay routing is difficult. Deeply understanding overlay network is necessary to solve the problem.

## 6.1 Overlay Network Layers

Depending on different responsibilities, the Internet can be modeled into four layers: Link, Network, Transport and Application. An overlay network is a computer network which is built on top of another network. Nodes in the overlay can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. P2P systems are overlay networks

running on top of the Internet.

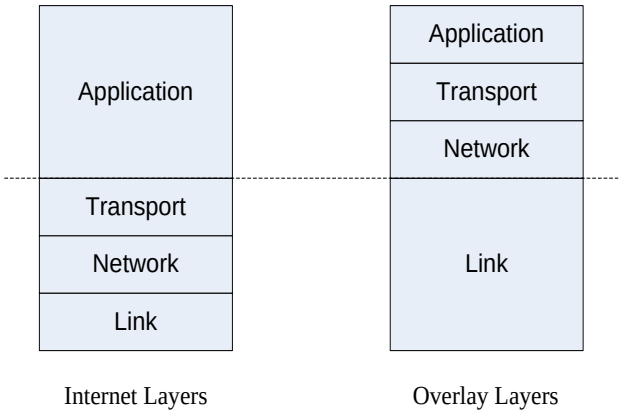


Figure 6.1: Layers of overlay network

The four layers model is essential for every network since the four layers are basic components for a working network. Based on their different responsibilities, overlay networks can also be divided into four layers. As figure 6.1 shows, the Link layer (bottom-most) of an overlay network includes all three bottom layers of network. This is because overlay network is connected by a logical link provided by network transport protocols, such as TCP or UDP, and those services also involve Network and Link layers. The Network layer of overlay network belongs to the Application layer of the Internet. Based on virtual connections provided by the Internet, overlay networks form their own network layer. With their particular routing protocols, such as Chord [5], Pastry [6], Gnutella [2], and KaZaA [3], the overlay network provides routing service, creating logical paths for transmitting data from node to node. The other two layers of overlay network are straight forward. The Transport provides a flow of data between two hosts for the application layer above. The application layer handles the details of the particular application. Both of those layers are also parts of the Application layer of the Internet.

## 6.2 A Common Overlay Network Under-layer

Since most of layers of the overlay networks belong to an application layer of the Internet, they ignore the underlying physical network. Thus, their routing is always sub-optimized. Chord in its original design, for instance, does not consider network proximity at all. As a result, its protocol for maintaining the overlay network is very light-weight, but messages may travel arbitrarily long distances in the Internet in each routing hop.

Because of the importance of network proximity, each overlay network developed its own technique to acquire network-aware information, such as RTTs and widely used landmark techniques as mentioned earller. In fact, the Network layer of overlay network is made up of two sub-layers. As figure 6.2 shows, the upper sub-layer called *routing* layer will be in charge of overlay network routing, and the lower one, with the name *network-proximity* layer, will provide network-aware information for above routing layer.

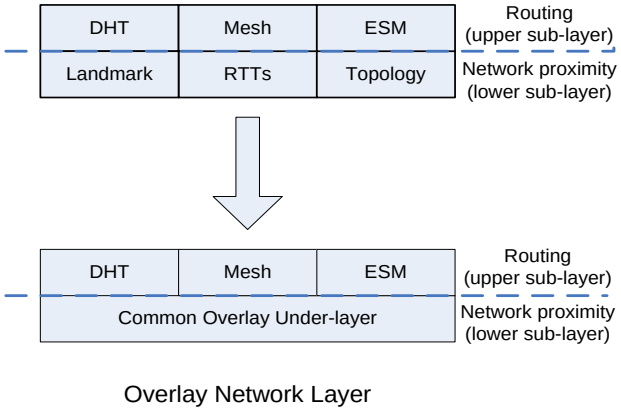


Figure 6.2: A common overlay network under-layer

Currently, each overlay network maintains its own network proximity system, such as landmark system for structured DHTs designs, RTTs networks for unstructured meshes or network-topology discovery for End-System Multicast (ESM) [47]. Although those techniques differ in details, they share the same network probe methods. For example, similar

*ping* methods are used to measure RTTs between peers within landmark systems, and variant *traceroute* tools are exploited for network topology. As a result, the same probe is preformed by different systems. Not only does the current pattern waste a lot of network resources, but it also generates a considerable amount of traffic due to the large number of peers. Thus, we proposed to build a common overlay network under-layer to provide network proximity information for all above overlay networks, just as figure 6.2 shows. That is to say, all overlay networks, no matter their different routing and application protocols, will build on a common under-layer, which includes all peers of different overlays and provides them network-aware information as requirements. The overlay technique in chapter 3 will be utilized to manage all peers. Also, the SNSA technique in chapter 2 will be exploited by providing network proximity information. As a result, not only will this common under-layer save network resources and reduce overlay overhead [48], but it will also provide more accurate network-aware information compared to any single overlay network.

## 6.3 Requirements of Different Overlay Networks

Before describing the common under-layer in detail, requirements of different overlay networks must be investigated.

### 6.3.1 Structured P2P Overlays

Currently, three major topology-aware routing techniques have been exploited by structured P2P systems: *proximity routing*, *topology based node ID assignment*, and *proximity neighbor selection* [49].

With *proximity routing*, the overlay is constructed without regard for the physical network topology. The technique exploits the fact that when a message is routed, there are potentially several possible next hop neighbors that are closer to the message's key in the

ID space. The idea is to select, among the possible next hops, the one that is closest in the physical network or one that represents a good compromise between progress in the ID space and proximity. The technique has been used in Chord and CAN. In order to support this, the network proximity sub-layer showed in figure 6.2 should provide estimation of latency between host and several possible next hops.

*Topology-based node ID assignment* attempts to map the overlay's logical ID space on the physical network so neighboring peers in the ID space will be close in the physical network. The technique has been successfully used in CAN, and has achieved low delay stretch results [50]. The notion of this technique is similar to the construction of our overlay mentioned earlier. To fully support this technique, not only the topology of the physical network is needed, but also the node density within each network unit is required. Otherwise, the constructed overlay would destroy the uniform population of the ID space, causing load balancing problems.

*Proximity neighbor selection*, also constructs a topology-aware overlay. However, instead of biasing the node ID assignment, it chooses routing table entries to refer to the topologically nearest among all nodes with a node ID in the desired portion of the ID space. The success of this technique depends on the degree of freedom an overlay protocol has in choosing routing table entries without affecting the expected number of routing hops. In prefix-based protocols, such as Tapestry and Pastry, the upper levels of the routing table allow great freedom in this choice, with lower levels leaving exponentially less of a choice. As a result, the expected delay of the first hop is very low; it increases exponentially with each hop, and the delay of the final hop dominates. Although, this technique seems different from the *proximity routing*, they essentially exploit the same network-aware information: estimation of latency between peers. The difference is that proximity routing uses latency information to select next routing hop, while proximity neighbor selection uses latency to construct routing tables.

### 6.3.2 Unstructured P2P Overlays

In general, three types of approaches have been proposed to improve search efficiency in unstructured P2P systems: *forwarding-based*, *cache-based*, and *overlay optimization*.

In forwarding-based approaches, instead of relaying the query messages to all its logical neighbors except the incoming peer, a peer selects a subset of its neighbors to relay the query. Each peer maintains some statistical information based on metrics, such as the number of results returned or the latency of the connection. A peer selects a subset of the neighbors, such as neighbors that have low latency, to send its query [51, 52]. For support of this kind of techniques, estimated latency between peers should be enough.

The cache-based approaches include data index caching and content caching. For example, KaZaA utilizes cooperative superpeers, each of which is an index server of a subset of peers. Queries are intended to forward to the nearest superpeers for quick solution. Essentially, this technique needs a network-proximity layer to provide latency information to its neighbors.

The third approach is based on overlay topology optimization [53, 54]. Studies [27] have shown that only 2 to 5 percent of Gnutella connections link peers within a single AS. However, more than 40 percent of all Gnutella peers are located within the top 10 ASes. This means that most Gnutella-generated traffic crosses AS borders. Because of the seriousness of topology mismatching, authors in [53, 54] proposed to use partial network-topology information to improve unstructured overlays. In order to reduce network traffic, they tend to favor peers within given router-hops, say two, and to cut crossing borders links. Those techniques need local router-level network topology information.

### 6.3.3 End System Multicast

End system multicast, Narada, is proposed in [47], which first constructs a rich connected graph on which to further construct shortest path spanning trees. Each tree is rooted at the corresponding source using well-known routing algorithms. To build and improve this kind of overlay network, not only the global Internet topology but also estimated network bandwidth between peers is needed.

Beyond all requirements above, some other issues are also crucial to overlay networks. The first one is membership of peers. Since peers within overlay networks tend to join and leave very frequently, the availability of peers is crucial for overlay routing. Connecting a failed peer will considerably increase routing latency and involve penalty in replacing the failed one in the routing table.

The second issue is the NAT traversal problem. Due to wide deployment of firewalls and NATs, connections between peers behind different firewalls are difficult. That is to say a peer may have lots of neighboring peers, but few of them are reachable. Many techniques exist, such as Simple Traversal of UDP over NATs(STUN) [55] and Traversal Using Relay NAT(TURN), but no technique works in every situation since NAT behavior is not standardized. Many techniques require a public server on a well-known globally-reachable IP address. Some methods use the server only when establishing the connection (such as STUN), while others are based on relaying all the data through it (such as TURN). As a result, to attain a peer's network connectivity information and find a public server for connection is vital for overlay routing.

The last one is security. Each peer is an important component of its overlay. Forged peers will significantly undermine the whole structure. The Sybil attack is one kind of such a threat, in which an attacker subverts a P2P network by creating a large number of pseudonymous entities, using them to gain a disproportionately large influence. Thus,

the overlay should have a mechanism to provide basic identification service of peers.

## 6.4 Design of the Common Under-layer

Although different overlays have their specific requirements, they can generally be categorized into two classes. One class is physical network related information, such as network topology information or latency estimation of given node. The other one is overlay peers related information, such as peer's availability, bandwidth or connectivity.

### 6.4.1 Network Related Primitives

Two major kinds of information are required for physical network. The most frequently used one is latency estimation, which includes several varieties, such as finding the nearest neighbors to a peer or the latency between given peers. The other one is network topology. Different overlays may require network topology information at a specified resolution and scope. Thus, we propose two primitives for common under-layer to support:

1. For network topology primitive, it should provide a graph of known network connectivity at specified resolution, such as ASes, or routers, and scope, such as the Internet, some AS, or everything within a radius of N hops.

2. For latency primitive, it should report network facts between a pair of peers according to a specified metric, such as AS hops, router hops, or measured latency.

In order to support topology primitives at AS resolution and scope, the common under-lay should keep an AS-level Internet topology graph. As chapter 2 mentioned, the Internet has a very clear topology graph at this level, and many public services provide that information, such as the CIDR Report [25] and the IRR [26]. However, to provide the Internet topology at the router-level is not easy. It needs either the privileged access

to BGP border routers or a huge amount of probing. Not only is neither of these methods an appropriate way for overlay network, but also router-level topology is overkill in most of time. Since all overlay peers are organized by our overlay technique mentioned in chapter 3, our SNSA technique in chapter 2 can be naturally exploited to provide scalable router-level topology for the upper layer. That is to say, for top ASes, the detailed router-level topology will be measured and kept. For middle ones, instead of full router-level topology, team-level will be provided. For small ones, only AS-level topology will be presented.

For latency primitive, topology information will be used to give the estimated latency at metric of AS and router hops. For latency metric, instead of periodically measuring RTTs to every peer, only router-level latency will be maintained. However, the number of probes is also very large even at router-level. In order to keep the cost to a minimum, RTTs will not be measured by common under-layer. Routing operations performed by overlay networks provide ample RTTs samples for all peers. By collecting the information, common under-lay can provide an accurate latency between the local router and all other routers. Also, this technique can be extended to supply latency between any pair of peers. For example, peer $X$ wants to know the latency from $A$ to $B$. By contacting the leader of $A$, $X$ can get the most recently measured latency from $A$ to $B$. Not only does this method eliminate duplicate probes, but it also makes all overlays cooperate together.

### 6.4.2 Overlay Peer Related Primitives

Beyond network related information, the common under-layer should also provide peer related information for an overlay network. Due to deployment on the open Internet environment, distribution of peers' resources and behaviors is highly skewed. Thus, the common overlay should provide the three kinds of information for overlay networks, namely resource, behavior, and security.

As analyzed in chapter 1, not every peer is suitable to grant regular responsibility

of the overlay, such as routing. Thus, some peer related information, such as availability, network connection type and so on, is crucial for overlay network. Security is also an important issue for overlays.

Two major kinds of information are required for physical network. The most frequently used one is latency estimation, which includes several varieties, such as finding nearest neighbors to a peer or the latency between given peers. The other one is network topology. Different overlays may require network topology information at a specified resolution and scope. Thus, we propose two primitives for common under-layer to support:

1. For peer resource primitive, it should provide a peer's basic resource information, for example CPU, memory, storage usage, network connection type(such as dial-up, DSL/cable modem and so on), and connectivity property.

2. For behavior primitive, it should report a peer's availability information. Both overlay(ID-based) and network(IP-based) availability of a peer need to be measured and reported, since a peer may run several overlay instances at the same time or just inconsecutively run some of them.

3. For security primitive, it should provide the ability to verify a peer's ID or IP, and ID-IP pair. That information is crucial to detect Sybil and ID swapping attacks.

In order to support resource and behavior primitives, each peer is designed to send a heartbeat message, which includes the name of overlay protocol, overlay ID, network, CPU and memory usage information, to its team leader. By this simple mechanism, the common under-layer can collect a peer's resource information and measure its availability at the same time. The connectivity can be checked with the help of leaders within a nearby AS. For security primitive, the techniques mentioned in chapter 5 can be exploited. For example, the net-print can be easily used to verify the ID-IP pair of a peer.

Part of the work within the chapter was published in [46].

# Chapter 7

# Evaluation

In this chapter, we evaluate our designs by simulation with the genuine Internet AS-level topology and latency and compare the results with current designs.

## 7.1 Experiments for the Structured Design

### 7.1.1 Experimental Setup

The genuine Internet latency used in our simulation was from the King method [56], which measured the network latencies of more than 1,700 DNS servers. The AS-level topology graph was from the CIDR Report [25]. By mapping each DNS server used in King to its AS, a 1701 nodes (ASes) graph with network latency and topology has been built. The average round trip delay between nodes pairs is 168ms, and the average AS path length is 2.97 hops. The p2psim [57] was used to simulate the Chord protocol with Proximity Neighbor Selection (PNS). A similar simulator of our overlay is developed under Java. All experiments were performed in a Dell Dimension PC with one 2.8GHz Pentium IV processor and 1.2GB RAM.

In order to make our experiments close to the deployment environment, the density

and distribution of nodes of real P2P overlay were needed. Recent research [12] has shown that the average nodes within one AS are 200 and 60 for KaZaA and Gnutella, respectively. Also the distribution is highly skewed and exhibits heavy tails, which can be approximated by Zipf's distribution. Thus, we use the Zipf distribution in our experiments. The average node density in our experiments is chosen to be 100 per AS, which is between the value of KaZaA and Gnutella. The system is modeled to have 10,000 nodes, which spread over 100 ASes under the Zipf distribution. These ASes are carefully selected from the original 1701 ones to keep the similar average latency and AS path length. For the Zipf distribution, the largest AS has 1,000 nodes and the smallest one has 36 nodes. Also, the uniform distribution has been simulated as the comparison.

We used the metrics of lookup latency stretch and failure rate to compare the performance of different overlays, and we used the bandwidth per node and link stress to evaluate their overhead. In the simulation, each node alternately crashed and re-joined the network; the interval between successive events for each was exponentially distributed with a mean time from 15 minutes to 1 hour. Each time a node joined, it was treated as a new one. Each node issues lookups for random keys at intervals exponentially distributed with a mean of 10 seconds. The p2psim was configured with a successors number of 16, base of 16 and refreshing intervals of 1 minute for both successors and fingers. This is a modest configuration without favoring either request successful rate or consumed bandwidth, according to the research of Li et al. [57]. For the simulation of our overlay, each group is configured to have 3 agents, the average group size is 9 teams, the average team size is 12 nodes, and each message can piggyback up to 3 agents' IP addresses. The size of a message is counted as 20 bytes (for packet overhead) plus 4 bytes for each IP address or node identifier mentioned in the message. When an agent leaves the overlay, the leader with the longest session time in the group will be upgraded to become an agent, and its backup will the leader of the team. All simulations ran 4 hours of simulated time, and statistics were collected only during the

last two hours.

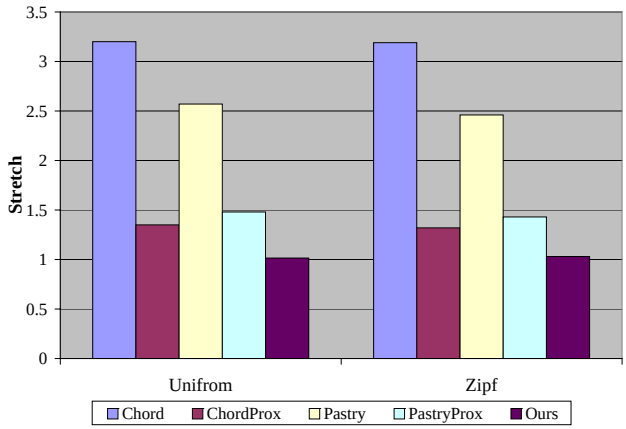


Figure 7.1: Overview of comparison between different structured P2P systems

### 7.1.2 Experimental Results

As a comparison between our system and others, we have compared with two other popular systems, Pastry and Chord, and their latest network proximity variations. As figure 7.1 shows, compared with the original version, the network proximity variations of both Chord and Pastry get significantly improved on their network latency stretch, which is defined at the ratio of overlay routing latency and the Internet routing one. The latency stretches reduce from 2.5 to about 1.5, which proves that to building the overlay close to the physical network is a promising way to optimize overlay systems. Also, as the figure indicates, all results under Zipf distraction are slightly better than uniform ones. This exhibits the potential of network locality of overlay peers. By fully exploiting physical network structure and network locality of overlay peers, our design is very close to the ideal result.

Figure 7.2 shows the result of the comparison of the lookup latency stretches of the two systems under churns. The extent of system churn is showed on the x-axis, which

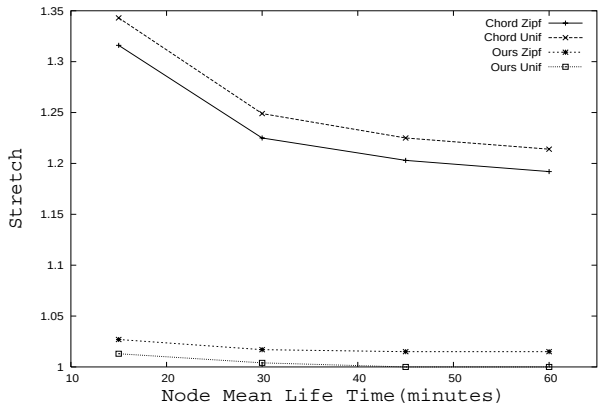


Figure 7.2: Stretch comparisons, under different system churn rates and distributions

is represented as the mean time of node life time. The shorter the mean life time, the more dynamic the system is. The stretch is the ratio between the latency of the overlay routing and the ideal Internet one. All protocols time out individual messages after an interval of three times the round-trip time to the target node, and the message is considered a failure. The stretches calculated in Figure 7.2 include both successe and failures. As the figure reveals, the stretches of Chord protocol increase quickly with the extent of churn; however, our overlay is not very sensitive. By exploiting more network locality from the skewed distributions of overlay peers, the stretches of Chord under Zipf are significantly better than the uniform ones. However, they are far from our system. By fully exploiting the physical network structure and the locality of peers, the stretches of our overlay are nearly the ideal ones. The results under the uniform are slightly better than the Zipf ones, since they do not involve communication across different ASes.

Figure 7.3 presents the request failure rate of different protocols under various environments. Because the symmetric protocol tries to involve every node into the routing operation, transitory nodes significantly compromise the performance. As figure 7.3 shows, the system churn is a significant impact to current structured P2P systems like Chord.

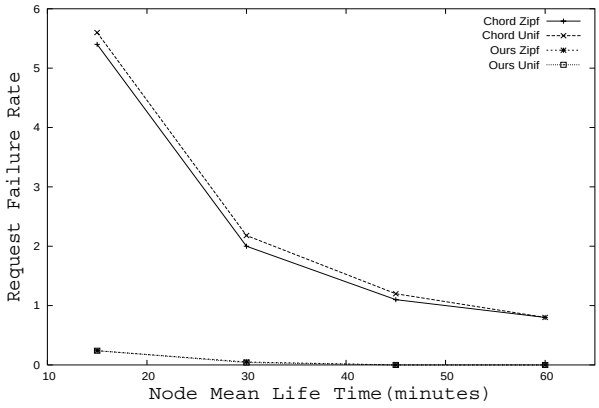


Figure 7.3: Request failure rate comparisons, under different system churn rates and node distributions

Although the Chord with PNS can make a little benefit from the skewed distribution of nodes, its failure rate rapidly raises to more than 5% under a mean node life time of 15 minutes. For our overlay, since only more stable nodes are assigned core operations, most of the requests are successfully delivered even under highly dynamical environments. Moreover, the gossip-style piggyback mechanism has proved to be very efficient to update information of agents. As a result, the failure rate is no more than 0.3% even under mean node life time of 15 minutes. Since the failure rates of our overlay are much better than the ones of Chord under all environments, we argue that granting equal responsibility to highly diverse resources and behaviors nodes may be a significant impact to the overlay. Paradoxically, an asymmetric protocol may be more appropriate for P2P applications.

Figure 7.4 shows the consumed bandwidth of nodes by different protocols and roles under Zipf distribution. The results of uniform are ignored for similarity. As an asymmetric protocol, the bandwidth of agents and the whole overlay are illustrated at the same time. As the figure reveals, frequent joining and leaving of nodes brings significant overhead to Chord overlay, while our overlay is not sensitive with those nodes. Moreover, even under symmetric protocol, the bandwidth consumed by each node varies largely because of randomly assigned

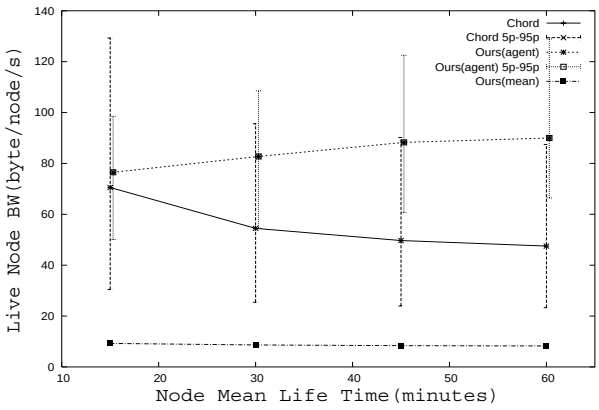


Figure 7.4: Bandwidth (mean and 5 to 95 percentiles) consumed by different protocols and roles, under system churn with Zipf distribution. The datum of our overlay is shifted right 0.3 to make figure clear.

node IDs and diverse session times. As the figure shows, the consumed bandwidth of agents in our overlay is significantly larger than the average value of symmetric overlay, like Chord. However, the top 5% nodes of Chord consume more or about the same close bandwidth as the agents, while the agents are only about 3% of our overlay. Furthermore, the average bandwidth in our overlay is less than one-fifth of the Chord. The rising of agent bandwidth with the falling of the churn is caused by the increased successful requests since more queries are solved.

P2P overlays are notorious for generating large amounts of Internet traffic. While the bandwidth of each node indicates the traffic per node, the traffic of the whole overlay can not be reflected. By mapping each node to its exact AS and finding out the paths between them, we can evaluate the traffic impact to the Internet back bones. By recording each link used by every message, the link stress can be calculated. Figure 7.5 shows the link stresses of different protocols under various environments. Both the total overlay link stress and the lookup (useful) one are shown. As the figure for Chord illustrates, system churn significantly increases P2P traffic to the Internet. Moreover, it reveals that the Chord overlay is not efficient. The useful (lookup) traffic is about 40% of the total traffic, and this ratio is

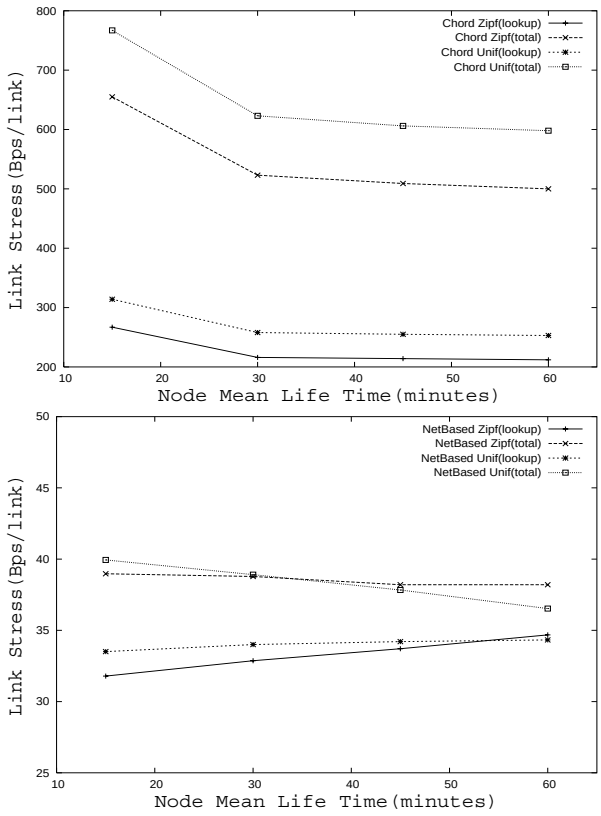


Figure 7.5: Link stresses put by different protocols to the Internet back bone, under different system churn rates and node distributions. The left one is for Chord protocol, the right one is for network based one.

independent of churn rates. On the contrary, our overlay is much more efficient. The useful part is more than 80% of the total one, and it keeps rising with the decreasing of the churn rate. Also, this experiment proves that network locality is very important for P2P overlays. As the figure for Chord shows, by exploiting the network locality under Zipf distribution, the traffic is 20% less than the one under uniform distribution. Moreover, by fully exploiting network locality of overlay nodes under our overlay, most of the communications happens within one AS or between nearby ones, only the necessary messages will be send out. As a result, lots of Internet backbone traffic is saved, and the total traffic of our overlay is less than one-tenth of Chord.

| Node capacity | Percentage of nodes |
|---|---|
| 5 | 20% |
| 50 | 45% |
| 500 | 30% |
| 5000 | 4.9% |
| 50000 | 0.1% |

Table 7.1: Gnutella-like node capacity distributions

## 7.2 Experiments for the Unstructured Design

### 7.2.1 Experimental Setups

The scalability is a serious problem for Gnutella-like unstructured P2P systems. When faced with a high aggregate query rate, Gnutella nodes quickly become overloaded and the system ceases to function satisfactorily. Furthermore, this problem gets worse as the size of the system increases. Based on prior research[58, 51, 9], Chawathe *et al.* [39] have proposed the Gia network, which tried to make Gnutella-like P2P systems scalable by topology adaptation, active flow control, one-hop replication, and biased random walks. Moreover, they provided an important experimental method to study scalability of unstructured systems. Our experiments adopt this method and use similar simulation parameters for comparison.

Like Chawathe's work, our simulation imposes capacity constraints on each of the nodes to capture the effect of the query load on the system. Each node $n$ is assigned a capacity $C_n$, which represents the number of messages that can be processed per unit time. We adopt the same capacity distribution with Chawathe, which has five levels as shown in Table 7.1. Also, each node $n$ is assigned a query generation rate $q_n$, which is the number

of queries that node $n$ generates per unit time. Of course, query rate $q_n$ of each node is bounded by its capacity $C_n$. Queries are modeled to search for specific keywords. Each keyword maps to a set of files, which is randomly replicated on nodes. All files associated with the specific keyword are potential answers for a query with the keyword. The term *replication factor* is used to refer to the fraction of nodes at which answers to queries reside.

A 10,000 nodes system has been simulated. The nodes are distributed in 100 groups. A team is set to have 10 nodes, and a group has 10 teams. The nearby ten groups form an area. The nodes with maximum capacity among the area become hub agents. Also, the maximum capacity nodes become the agents of a group and the leader of a team, respectively. As the search method of an expanding ring, each node will try to solve the query at a larger scope than the previous one. The normal searching sequence is from regular node to a leader, then to an agent, to a hub agent, next to nearby hub agents, and finally to all groups. When a leader is overloaded, the node can send the query directly to its agent. A penalty of three times RTT will be added to time out an overloaded hop.

We look at three aspects of the system's performance as a function of the offered load: the *success rate* measured as the fraction of the queries that successfully locate the desired files, the *hop count* as the number of hops to reach the requested files, and the *latency* as the time taken by a query from start to finish. The system is simulated under varying replication factors and query rates. The number of responses for a query is set to be one, and the TTL is 10. The experiments for multiple responses are not performed, since previous research [39] has shown that the performance of a query for $k$ responses at a replication factor of $r$ is equivalent to that of a query for a single response at a correspondingly lower replication factor of $r/k$.

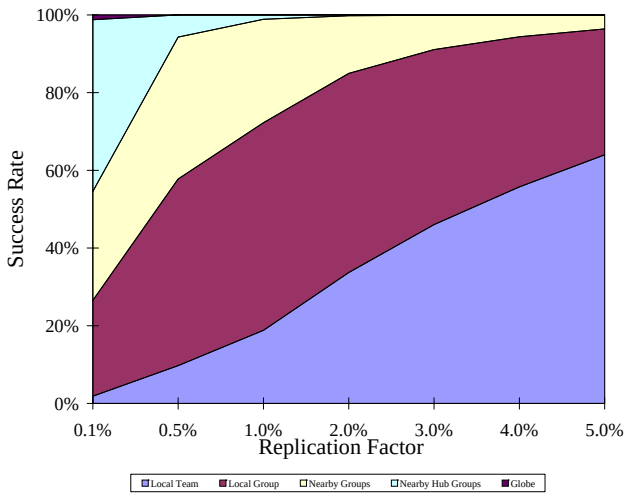


Figure 7.6: Expanding ring search for unstructured design. Shown is the search success ratio of different search range under various replication factors.

### 7.2.2 Experimental Results

Figure 7.6 presents the impact of the expanding ring search of our unstructured design. Our expanding ring method guarantees that search area increases with hop. The search area(success rate) is exponential with hops/messages. As the figure shows, the area covered by local team(1 hop/message) is proportion-like, which means power is equal to one. The area curve is quadratic for local group(2 hops/message), and cubic for nearby groups(3 hops/message). Compared with a classical unstructured system such as Gnutella, its query can reach an exponential number of hosts with hops. However, not only are quite a few of the hosts duplicated, but also the flooding method generates an exponential number of messages, which involves a serious problem of scalability.

Figure 7.7 shows the results of success rate, hop-count and latency under increasing query load. Similar to the results of the Gia network, we notice a knee point in the curves beyond which the success rate drops sharply and latency increases rapidly in each figure. The hop-count holds steady until the knee point and then decreases. This is because hop-count is measured only for successful queries. Under an increasing load, queries tend to be

solved by nodes within a few hops from the originator. The *Collapse Point (CP)* is defined by the per node query rate at the knee, beyond which the success rate drops blew 90%. This metric reflects the total system capacity. The *Hop-count before collapse (CP-HC)* is the average hop-count prior to collapse. The latency is not retained as a metric since it is effectively captured by the collapse point. Compared with the Gia network, our system is slightly better in both CP and CP-HC, especially for a lower replication factor. The CP of our system under 0.1% replication factor is 20, while that of the Gia network is 7. The major result of this is that our system has a much lower CP-HC. The CP-HC of our system is 3.48, while the one of Gias' is more than 15.

Ideally, we expect a system to archive a high success rate while maintaining a low hop-count and delay. In fact, the key to archive this is to lower the hop-count of queries, or, in other words, to quickly find the answers. In order to quickly find the answers, current unstructured search protocols [39, 9] tend to exploit the hierarchy of capacity and shared contents of peers. Queries are deliberately led to the nodes with a higher capacity and/or more content in order to get results faster. Although those protocols significantly improve system performance compared with original flooding protocol, they have their limitations. Due to random connections among peers, the circular forwarding is unavoidable. A query may arrive at a certain node more than once, which increases hop-count and wastes recourse. Although those protocols ensure to expand search scope hop by hop, the expanding scope is overlapped since two high degree nodes likely connect the same portion of nodes. By avoiding the problems above, the expanding ring protocol of our system successfully archives a high success rate with a low hop-count.

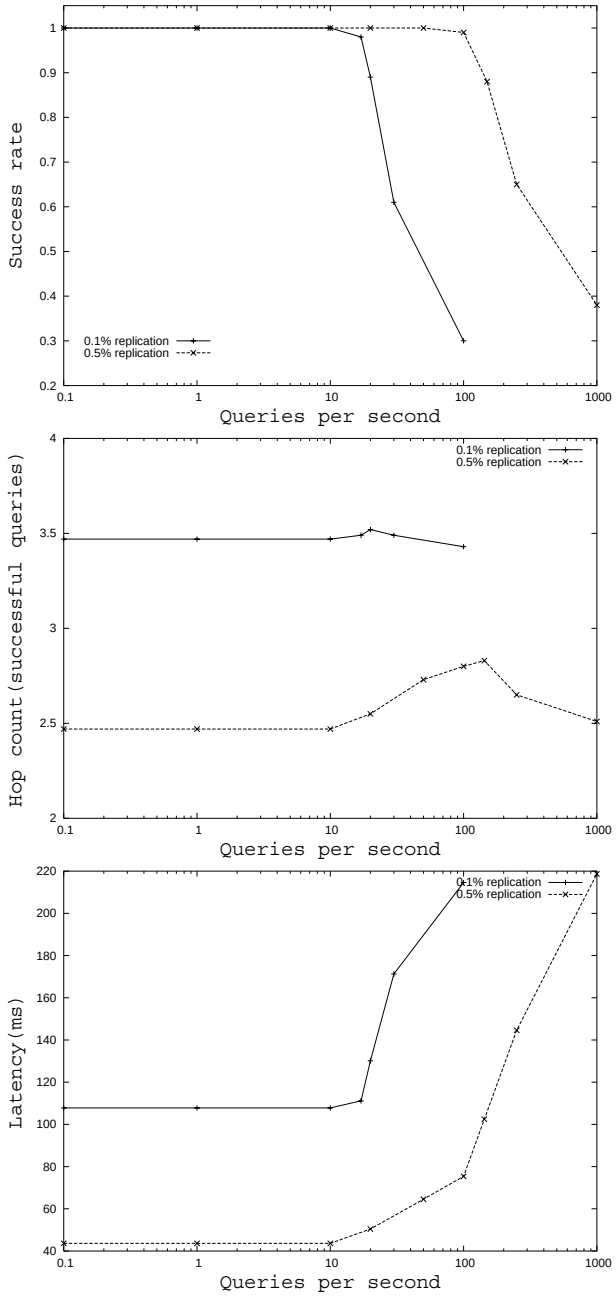


Figure 7.7: Keywords search for the unstructured P2P design. Shown is the search success rate of different search range under various replication factors. The system has 10,000 nodes, which distributed in 100 ASs, and the average team size is 10.

# Chapter 8

# Conclusions and Future Work

## 8.1 Conclusions

From a different perspective, this dissertation proposes a new approach to building a highly efficient and secure P2P overlay by fully exploiting the physical network structure of the Internet and the network locality of overlay peers. In addition to the network latency between peers, a traditional measurement for network locality of distributed systems, two other important network properties, bandwidth and connectivity between peers, are introduced to the network locality of P2P systems. Moreover, a novel SNSA technique has been developed to efficiently discover and exploit the network locality of overlay peers. Based on the network locality, a unique P2P overlay, which can support both structured and unstructured systems, is constructed. Based on the observation that every peer must subject to the network configuration and administration imposed by ISPs, a novel identifier, net-print, is proposed to identify each peer under a P2P environment. Based on the SNSA technique and the net-print, a distributed authentication and secure routing mechanisms are developed under a P2P environment. Beware of the importance of network proximity. Each overlay network developed its own technique to acquire network-aware information.

As a result, the same probe is preformed by different systems. This pattern not only wastes a lot of network resources, but it also generates considered traffic due to the large number of peers. A common overlay network under-layer to provide network proximity information for all above overlay networks is proposed.

By fully exploiting the physical network structure of the Internet and the network locality of overlay peers, many difficulties faced by current P2P systems, such as the scalability and searching overhead problems for unstructured ones, and efficiency and security problems for structured ones, are better solved in our design. We believe that makeing full use of the physical network structure of the Internet and the network locality of overlay peers is a promotion way toward P2P and other large-scale distributed applications.

## 8.2 Future Work

### 8.2.1 A Hybrid P2P System

Although our structured and unstructured P2P systems are based on the same overlay structure, there are still two different systems. In fact, both of them have their advantages and disadvantages. For structured P2P systems, by exploiting elaborate overlay structure and DHT routing, they are more scalable, and highly efficient and accurate with locating objects within a very large system. For unstructured ones, based on randomly connected overlay and gossip-style routing, they are simple and friendly on keyword search, lower maintenance overhead, and robust facing system churn or even network disaster.

Based on our unique overlay structure, which fully exploits the network structure of the Internet and the network locality of overlay peers, a hybrid P2P system with advantages of both structured and unstructured ones can be built in the future. The search methods of the structured and the unstructured ones are perfect supplements to each other. For well-duplicated objects, such as popular music, unstructured-style search can be used. In

addition, for unpopular objects, structured-style search can be exploited. While regular overlay structure for structured systems can facilitate overlay functions, such as routing and lookup, the randomly connected structure of unstructured systems is very robust under system churn or even network disaster. Thus, we expect to make our hybrid system more efficient and robust by exploiting the characteristics of both regular structures and random ones.

### 8.2.2 Other Security Issues for P2P Systems

In chapter 5, we discuss securing the routing of our structured P2P system. Although secure routing is not a serious problem for unstructured P2P systems, they still face many other secure problems under P2P environment. Some examples of frequent attacks in P2P systems could be: malware in the P2P system itself, poisoning/polluting attacks (provide files whose advertised description and actual content are different), denial of service attacks (which will make the network or certain parts of it break completely, or only run very slowly), the insertion of viruses to carried data, defection attacks/free riding (users that make use of the network without contributing resources to it), identity attacks, or spamming.

Based on the accurate physical network information provided by the SNSA technique, the *net-print* is self-certifying, and can be exploited to build a fully distributed authentication mechanism under the environment without mutual trust. Based on this distributed authentication mechanism, each node can be identified. Although the identification of nodes can not directly thwart most attacks, it is helpful in establishing a trusted reputation system within a P2P system. By recording, warning, or even punishing faulty nodes, it will be more and more difficult for malicious attacks to occur.

# Bibliography

[1] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-law Relationships of the Internet Topology," in *Proceedings of the 1999 conference on applications, technologies, architectures, and protocols for computer communications of SIGCOMM'99*, Cambridge, MA, 1999, pp. 251–262. [Online]. Available: citeseer.nj.nec.com/faloutsos99powerlaw.html

[2] Gnutella, "Gnutella hosts," http://www.gnutellahosts.com.

[3] KaZaA, "KaZaA Media Desktop," http://www.kazaa.com.

[4] Gnutella2, "Gnutella2 Developer' Network," http://www.gnutella2.com.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, San Diego, CA, 2001, pp. 149–160.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, Nov. 2001. [Online]. Available: http://www.research.microsoft.com/ antr/PAST/pastry.pdf

[7] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," UC Berkeley, Tech. Rep. UCB/CSD-01-1141, Apr. 2001. [Online]. Available: citeseer.nj.nec.com/zhao01tapestry.html

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," in *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, San Diego, CA, 2001. [Online]. Available: citeseer.nj.nec.com/ratnasamy01scalable.html

[9] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, January 2002.

[10] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *Proceedings of the USENIX Annual Technical Conference*, 2004.

[11] ——, "Handling churn in a dht," University of California, Berkeley, Tech. Rep. UCB//CSD-03-1299, December 2003.

[12] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks," in *In Proc. ACM SIGCOMM Internet Measurement Workshop, Marseille, France, Nov. 2002.*, 2002.

[13] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proceedings of the 23st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-04)*, 2004.

[14] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *USENIX First Symposium on Nerworked Systems Design and Implementation (NSDI'04)*, Mar. 2004.

[15] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding Availability," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, 2003.

[16] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, Dec 2002.

[17] J. R. Douceur, "The Sybil Attack," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2002.

[18] E. Sit and R. Morris, "Security Considerations for Peer-to-Peer Distributed Hash Tables," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2002.

[19] M. Yang, Z. Zhang, X. Li, and Y. Dai, "An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'05)*, 2005.

[20] RFC2453, "Routing Information Protocol (RIP) Version 2," http://www.ietf.org/rfc/rfc2453.txt.

[21] RFC2328, "Open Shortest Path First (OSPF) Version 2," http://www.ietf.org/rfc/rfc2328.txt.

[22] RFC1771, "A Border Gateway Protocol 4 (BGP-4)," http://www.ietf.org/rfc/rfc1771.txt.

[23] H. Chang, S. Jamin, and W. Willinger, "Inferring AS-level Internet topology from router-level path traces," in *Proceeding of SPIE ITCom 2001*, Denver, CO, August 2001. [Online]. Available: citeseer.nj.nec.com/chang01inferring.html

[24] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "The Origin of Power-Laws in Internet Topologies Revisited," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, vol. 2, Piscataway, NJ, June 23–27 2002, pp. 608–617.

[25] CIDR-Report, "The CIDR Report," http://www.cidr-report.org.

[26] M. Network, "Internet Routing Registry," http://www.irr.net.

[27] M. Ripeanu, I. Foster, and A. Iamnitchi, "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. [Online]. Available: citeseer.nj.nec.com/ripeanu02mapping.html

[28] Routeviews.org, "Route Views Archive," http://www.routeviews.org.

[29] Traceroute.org, "Public Route Server List," http://www.traceroute.org.

[30] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," in *Proceeding of the 23nd Internatinal Conference on Distributed Computing System(ICDCS03)*, 2003. [Online]. Available: citeseer.nj.nec.com/xu03building.html

[31] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, 6 2002.

[32] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in *Proceedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2004.

[33] RFC791, "Internet Protocol DARPA Internet Program Protocol Specification," http://www.ietf.org/rfc/rfc0791.txt.

[34] L. M., A. S., and F. A., "A Efficient Algorithms to Implement Unreliable Failure Detectors in Parially Synchronous Systems," in *Proceedings of the 13th Symposium on Distributed Computing (DISC'99)*, Bratislava, Slovaquia, 1999.

[35] F. 180-1, "Secure Hash Standard," U.S. Department of Commerce/NIST, National Technical Information Service, Apr. 1995.

[36] A. Gupta, B. Liskov, and R. Rodrigues, "Efficient Routing for Peer-to-Peer Overlays," in *First Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, Mar. 2004.

[37] H. Weatherspoon and J. D. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[38] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the Evolution of Peer-to-Peer Systems," in *Proceedings of ACM PODC*, July 2002.

[39] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2003.

[40] H. Wang, Y. Zhu, and Y. Hu, "To Unify Structured and Unstructured P2P Systems," in *Proceeding of the 19th International Parallel and Distributed Processing Symposium (IPDPS05)*, Denver, Colorado, April 2005.

[41] Y. Zhu, H. Wang, and Y. Hu, "Integrating semantics-based access mechanisms with p2p file systems," in *Peer-to-Peer Computing*, 2003.

[42] ——, "A super-peer based lookup in structured peer-to-peer systems," in *ISCA PDCS*, 2003, pp. 465–470.

[43] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *SIGCOMM*, 2006, pp. 267–278.

[44] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stocia, "Wide-area Cooperative Storage with CFS," in *Proceedings of ACM SOSP'01*, Oct. 2001.

[45] H. Wang, Y. Zhu, and Y. Hu, "An efficient and secure peer-to-peer overlay network," in *Proceeding of the 30th IEEE Conference on Local Computer Networks (LCN)*, 2005.

[46] H. Wang and Y. Hu, "Building a peer-to-peer overlay for efficient routing and low maintenance," in *EUC Workshops*, 2005, pp. 766–775.

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

[48] A. Nakao, L. L. Peterson, and A. C. Bavier, "A routing underlay for overlay networks," in *SIGCOMM*, 2003, pp. 11–18.

[49] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Topology-aware routing in structured Peer-to-Peer overlay networks," Microsoft Research, One Microsoft Way, Redmond, Tech. Rep. 82, 2002. [Online]. Available: citeseer.nj.nec.com/castro02topologyaware.html

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

[51] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," in *Proceedings of 16th ACM International Conference on Supercomputing (ICS-02)*, New York, NY, June 2002.

[52] Z. Zhuang, Y. Liu, L. Xiao, and L. M. Ni, "Hybrid periodical flooding in unstructured peer-to-peer networks," in *ICPP*, 2003, pp. 171–178.

[53] Y. Liu, X. Liu, L. Xiao, L. M. Ni, , and X. Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proceedings of the 23st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM'04)*, 6 2004.

[54] G. S. Manku, M. Naor, and U. Wieder, "Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks," in *STOC*, 2004, pp. 54–63.

[55] RFC3489, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," http://www.ietf.org/rfc/rfc3489.txt.

[56] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," in *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002.

[57] J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," in *Proceedings of 24th IEEE INFOCOM*, March 2005.

[58] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in Power-law Networks," in *Pyhsical Review E 64*, 2001.

[59] M. Freedman and D. Mazieres, "Sloppy hashing and self-organizing clusters," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, 2003. [Online]. Available: citeseer.nj.nec.com/freedman03sloppy.html

[60] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, "Brocade: Landmark routing on overlay networks," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2002. [Online]. Available: citeseer.nj.nec.com/zhao02brocade.html

[61] "AS Graph Data Sets," http://topology.eecs.umich.edu/data.html.

[62] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore, and k claffy, "Distance Metrics in the Internet," in *IEEE International Telecommunications Symposium(ITS) 2002*, 2002.

[63] B. Krishnamurthy, J. Wang, and Y. Xie, "Early Measurements of a Cluster-based Architecture for P2P Systems," in *ACM SIGCOMM Internet Measurement Workshop (San Francisco, Nov. 2001)*, San Francisco, CA, 2001. [Online]. Available: citeseer.nj.nec.com/krishnamurthy01early.html

[64] IRR, "List of Routing Registies," http://www.irr.net/docs/list.html.

[65] "GT-ITM," www.cc.gatech.edu/projects/gtitm.

[66] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Observations on the Dynamic Evolution of Peer-to-Peer Networks," in *Proceedings of IPTPS 2001*, 2001. [Online]. Available: citeseer.nj.nec.com/article/liben-nowell01observations.html

[67] RFC1772, "Application of the Border Gateway Protocol in the Internet," http://www.ietf.org/rfc/rfc1772.txt.

[68] RFC2365, "Administratively Scoped IP Multicast," http://www.ietf.org/rfc/rfc2365.txt.

[69] RFC1112, "Internet Group Management Protocol," http://www.ietf.org/rfc/rfc1112.txt.

[70] RFC2236, "Internet Group Management Protocol, Version 2," http://www.ietf.org/rfc/rfc2236.txt.

[71] "Chord Simulator," http://www.pdos.lcs.mit.edu/cgi-bin/cvsweb.cgi/sfsnet/simulator/.

[72] Z. Xu and Y. Hu, "SBARC:A Supernode Based Peer-to-Peer File Sharing System," in *Proceedings of the 8th IEEE Symposium on Computers and Communications (ISCC'2003)*, kemer - Antalya, Turkey, June 2003, pp. 1053–1058.

[73] L. Garces-Erice, K. W. Ross, E. W. Biersack, P. A. Felber, and G. Urvoy-Keller, "Topology-centric look-up service," in *Proceedings of COST264 Fifth International Workshop on Networked Group Communications (NGC)*, Munich, Germany, 2003. [Online]. Available: citeseer.nj.nec.com/564384.html

[74] A. Bakker, E. Amade, G. Ballintijn, I. Kuz, P. Verkaik, I. van der Wijk, M. van Steen, and A. S. Tanenbaum, "The Globe distribution network," in *Proceedings of the USENIX Annual Technical Conference*, 2000, pp. 141–152. [Online]. Available: citeseer.nj.nec.com/bakker00globe.html

[75] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," in *Proceedings of the 2000 conference on applications, technologies, rchitectures, and protocols for computer communications SIGCOMM*, 2000, pp. 97–110. [Online]. Available: citeseer.nj.nec.com/krishnamurthy00networkaware.html

[76] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Topology-aware routing in structured peer-to-peer overlay networks," Microsoft Research, One Microsoft Way, Redmond, Tech. Rep. 82, 2002. [Online]. Available: citeseer.nj.nec.com/castro02topologyaware.html

[77] ——, "Exploiting network proximity in Peer-to-Peer overlay networks," in *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, 2002.

[78] G. Ballintijn, M. van Steen, and A. S. Tanenbaum, "Exploiting location awareness for scalable location-independent object IDs," in *Proceedings of Fifth Annual ASCI Conference*, Heijen, The Netherlands, June 1999, pp. 321–328. [Online]. Available: citeseer.nj.nec.com/ballintijn99exploiting.html

[79] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Proximity Neighbor Selection in Tree-based Structured Peer-to-peer Overlays," in *International Workshp on Future Directions in Distributed Computing (FuDiCo)*, 2002.

[80] L. Zhou and R. van Renesse, "P6P: A Peer-to-Peer Approach to Internet Infrastructure," in *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, 2004.

[81] A. Gupta, B. Liskov, and R. Rodrigues, "One Hop Lookups for Peer-to-Peer Overlays," in *USENIX Annual Conference Workshop on Hot Topics in Operaring Systems(HotOS 2003)*, 5 2003.

[82] ——, "Efficient Routing for Peer-to-Peer Overlays," in *USENIX First Symposium on Nerworked Systems Design and Implementation (NSDI'04)*, Mar. 2004.

[83] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Network*, vol. 9, no. 6, pp. 733–745, Dec 2001.

[84] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation(OSDI02)*, Boston, MA, 2002.

[85] Y. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proceedings of the 23st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-04)*, Hong Kong, China, 2004.

[86] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A Distributed Approach to Solving Overlay Mismatching Problem," in *Proceeding of the 24nd Internatinal Conference on Distributed Computing System(ICDCS04)*, Tokyo, Japn, March 2004.

[87] Y. Liu, L. Xiao, and L. M. Ni, "Building a Scalable Bipartite P2P Overlay Network," in *Proceeding of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*, Santa Fe, New Mexico, April 2004.

[88] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, 2003.

[89] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, "Practical, distributed network coordinates," in *Proceedings of Second Workshop on Hot Topics in Networks(HotNets03)*, Nov 2003.

[90] T. E. Ng and H. Zhang, "A Network Positioning System for the Internet," in *Proceedings of the USENIX Annual Technical Conference*, Boston, MA, June 2004.

[91] R. Siamwalla, R. Sharma, and S. Keshav, "Discovering Internet Topology," www.cs.cornell.edu/skeshav/papers/discovery.pdf.

[92] Z. Xu, M. Mahalingam, and M. Karlsson, "Turning Heterogeneity into an Advantage in Overlay Routing," in *Proceedings of the 22st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-03)*, 2003.

[93] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling internet routers using optics," in *Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM03)*, 2003.

[94] A. T. Mizrak, Y. Cheng, V. Kumar, and S. Savage, "Structured Superpeers: Leveraging Heterogeneity to Provide Constant-Time Lookup," in *IEEE Workshop on Internet Applications*, 2003.

[95] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, 2003.

[96] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie, "HiScamp: self-organizing hierarchical membership protocol," in *Proceedings of the 10th European ACM SIGOPS workshop*, Sept. 2002.

[97] J. Chu, K. Labonte, and B. N. Levine, "Availability and Locality Measurements of Peer-To-Peer File Systems," in *Proceedings of ITCom: Scalability and Traffic Control in IP Networks*, July 2003.

[98] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File Sharing Workload," in *Proceedings of ACM SOSP*, Oct. 2003.

[99] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," in *Proceedings of the 2002 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, Feb. 2002.

[100] R. M. and T. F., "Group membership failure dectection: a simple protocol and its probabilistic analysis," *Distributed Systems Engineering*, vol. 6, no. 3, pp. 95–102, 1999.

[101] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. L. Peterson, and R. Wang, "Overlay mesh construction using interleaved spanning trees," in *INFOCOM*, 2004.