OUKA
Osaka University Knowledge Archive

Title	Constructing Proximity-Aware Balanced Overlay Networks
Author(s)	Makikawa, Fuminori
Citation	大阪大学, 2011, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/27638
rights	
Note	

Osaka University Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

Osaka University

博甲 14996

Constructing Proximity-Aware Balanced Overlay Networks

January 2011

Fuminori MAKIKAWA

Constructing Proximity-Aware Balanced Overlay Networks

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2011

Fuminori MAKIKAWA

Abstract

As the size of a network expands, peer-to-peer (P2P) systems increasingly become more important than traditional server/client systems. In a P2P system, each peer individually determines which peers it connects to with links. The virtual network thus constructed is called an overlay network. Peers communicate with each other through the overlay network. Two types of overlay networks are used in P2P systems: unstructured and structured. An unstructured overlay network dose not obey any rigid topology; therefore, peers can freely choose the peers to which they connect. This feature has low maintenance cost, but the ability of peers to search for required data items is limited. In contrast, a structured overlay network follows a particular topology. This topology constraint allows peers to reliably search for data items. However, maintaining the topology incurs some cost. For example, peers must send maintenance messages periodically. The different types of overlays thus have different advantages and disadvantages. However, search delay is one of the most important performance factors for both types. This dissertation addresses the problem of improving search delay for both unstructured and structured overlay networks.

First, I propose an approach for constructing unstructured overlay networks with low search delay and short paths. The basic concept behind the approach is

to repeatedly perform local topology changes so that peers can connect to those physically close to them, thus reflecting the proximity of peers in overlay construction. The existing methods that construct proximity-aware overlay networks keep links as short as possible; however, the problem of large path length between two distant peers arises. My approach solves this problem by letting each peer manage extra-long links in addition to short links. To maintain better control over the long links rather than setting them totally randomly, I associate an ideal length with them and keep their length close to the ideal. To demonstrate the usefulness of my approach, I incorporate it into the existing overlay construction methods and compare the overlay networks obtained with and without this approach. The simulation results show that my proposed approach exhibits significant improvement not only in search delay but also in other criteria such as clustering coefficient and tolerance to network partitioning.

Second, I propose a new method for constructing a skip graph, a well-known structured overlay network. In the original skip graph construction, peers choose their links randomly. This random approach may cause topological imbalance in the resulting graph, thus increasing search delay. In my construction, peers change the graph topology on the basis of proximity and balance, both of which significantly affect search delay. The proposed construction can be adapted to skip graph variants. In this dissertation, I examine two such variants. The simulation results show that my approach can construct skip graphs with lower search delay than in the original skip graph construction; this is also the case for the two skip graph variants.

List of Major Publications

[1] Fuminori Makikawa, Tatsuhiro Tsuchiya and Tohru Kikuno, "Constructing Overlay Networks with Short Paths and Low Communication Cost," IEICE Transactions on Information and Systems, Vol.E93-D, No.6, pages 1540–1548, June 2010.

[2] Fuminori Makikawa, Tatsuhiro Tsuchiya and Tohru Kikuno, "Balance and Proximity-Aware Skip Graph Construction," International Workshop on Advances in Networking and Computing (WANC), 2010.

[3] Fuminori Makikawa, Tatsuhiro Tsuchiya and Tohru Kikuno, "Constructing Skip Graphs with Proximity," IPSJ Technical Reports, Vol.2009-GN-73, No.5, pages 1–8, November 2009. (In Japanese)

[4] Fuminori Makikawa, Takafumi Matsuo, Tatsuhiro Tsuchiya and Tohru Kikuno, "Constructing Overlay Networks with Low Link Costs and Short Paths," Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007), pages 299–304, July 2007.

[5] Fuminori Makikawa, Tatsuhiro Tsuchiya and Tohru Kikuno, "On Decreasing the Number of Hops Needed for Long-Distance Communications in Overlay Networks," IEICE Technical Report, Vol.106, No.292• pages 13–18•

October 2006. (In Japanese)

Acknowledgements

During the course of this study, I have been fortunate to have received assistance from many individuals. I would especially like to thank my supervisor Professor Tohru Kikuno for his continuous support, encouragement and guidance.

I am also very grateful to the members of my dissertation review committee, Professor Takao Onoye and Associate Professor Hiroyuki Ohsaki, for their invaluable comments and helpful criticism of this dissertation.

I express my special thanks to the members of my advisory committee, Professor Yoshiaki Kakuda, Dr Shinichi Nagano and Associate Professor Hiroyuki Ohsaki, for their insightful comments and suggestions.

I also express my special thanks to Associate Professor Tatsuhiro Tsuchiya for his continuous assistance and helpful advice.

Finally, I thank many friends in the Graduate School of Information Science and Technology at Osaka University, who gave me much help.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Main Results 2

1.2.1 Constructing Unstructured Overlay Networks with Low Link Delay and Short Paths 2

1.2.2 Constructing Balanced and Proximity-Aware Skip Graphs 3

1.2.3 Overview of Dissertation 3

2 Overlay Networks **5**

2.1 Unstructured Overlay Networks 9

2.2 Structured Overlay Networks 9

3 Constructing Unstructured Overlay Networks with Low Link Delay and Short Paths **11**

3.1 Introduction 11

3.2 Related Work 13

3.3 Performance Measures 14

3.4 Existing Proximity-Aware Methods 17

3.4.1 Localiser 17

3.4.2 mOverlay 17
3.4.3 Problem with Traditional Proximity-Aware Methods . . . 19
3.4.4 GoCast 20
3.5 My Proposed Technique 21
3.5.1 Illustrative Example 27
3.6 Simulation Evaluation 27
3.6.1 Simulation Settings 27
3.6.2 Simulation 1: Parameters of Proposed Technique 28
3.6.3 Simulation 2: Comparison with Existing Methods 34
3.6.4 Simulation 3: Resilience to Peer Failure 40
3.7 Discussion 43
3.8 Summary 45

4 Constructing Balanced and Proximity-Aware Skip Graphs 47
4.1 Introduction 47
4.2 Skip Graphs 48
4.3 Balanced and Proximity-Aware Construction 51
4.3.1 Criterion of Structural Balance 51
4.3.2 Topology Reconstruction Algorithm 53
4.4 Required Messages 56
4.4.1 Searching Operation 57
4.4.2 Peer Joining 58
4.4.3 Peer Removal 59
4.4.4 Topology Reconstruction 60
4.5 Application to Skip Graph Variants 60

4.5.1 Skip B-tree 61
4.5.2 Rainbow Skip Graph 62
4.6 Simulations 64
4.6.1 Effect of K 64
4.6.2 Comparison of Search Delay and Search Path Length . . . 65
4.7 Discussion 68
4.8 Summary 72

5 Conclusion **73**
5.1 Achievements 73
5.2 Future Research 74

List of Figures

2.1 Physical Network(lower) and Overlay Network(upper) 6

2.2 Centralised Overlay Network 7

2.3 Distributed Overlay Network 7

3.1 Localiser Algorithm 18

3.2 mOverlay Network 20

3.3 GoCast Network 21

3.4 Random Overlay Network 24

3.5 Overlay Network Constructed by Localiser 25

3.6 Overlay Network Constructed by the Proposed Approach 26

3.7 Ratio of Long Links and Average Search Delay (p-Localiser) . . . 29

3.8 Ratio of Long Links and Average Path Length (p-Localiser) 30

3.9 Ratio of Long Links and Average Search Delay (p-mOverlay) . . 32

3.10 Ratio of Long Links and Average Path Length (p-mOverlay) . . . 33

3.11 Average Search Delay (Random, Localiser, p-Localiser, GoCast) . 35

3.12 Average Search Delay (mOverlay, p-mOverlay) 36

3.13 Average Path Length (Random, Localiser, p-Localiser, GoCast) . . 37

3.14 Average Path Length (mOverlay, p-mOverlay) 38

3.15 Random Failure Rate and Reachability (Random, Localiser, p-Localiser, GoCast) 41
3.16 Random Failure Rate and Reachability (mOverlay, p-mOverlay) . 42

4.1 Skip Graph 49
4.2 Well-Balanced Skip Graph 52
4.3 Imbalanced Skip Graph 52
4.4 Algorithm Outline 54
4.5 Search Path Length as a Function of K 66
4.6 Search Delay as a Function of K 67
4.7 Relationship between Balance and Number of Times Reconstruction is Executed 68
4.8 Comparison of Search Path Length 69
4.9 Comparison of Search Delay 70

List of Tables

2.1 Unstructured and Structured Overlay Networks 8

3.1 Clustering Coefficient, Path Length and Search Delay 39

Chapter 1

Introduction

1.1 Background

A peer-to-peer (P2P) application uses an overlay network, which is a virtual network constructed over a physical network. Because current P2P applications consist of a numerous number of peers, it is increasingly important to construct communication-efficient overlay networks.

In P2P systems, collections of data content are shared by all peers. Therefore, the most important property of overlay networks is the delay time required for searching the target peer or data. In this dissertation, I improve two properties of overlay networks to reduce search delay: path length and link delay.

Path length refers to the number of overlay links in a path between two peers. Clearly, path length should be short to reduce the number of relay peers that are required to forward a message and thus achieve efficient communication. The link delay between two peers is usually expressed as the distance between them in the physical network. If link delay is not considered in overlay construction,

considerable redundant traffic is produced, resulting in inefficient communication and performance degradation of P2P applications. This dissertation addresses these two issues in the construction of both unstructured and structured overlay networks.

1.2 Main Results

1.2.1 Constructing Unstructured Overlay Networks with Low Link Delay and Short Paths

As its first contribution, this dissertation proposes a new method for constructing unstructured overlay networks with low link delay and short paths.

To reduce the link delay, the existing approaches typically establish overlay links between two nearby peers in the physical network [17, 30]. However, this approach increases the path length between two physically distant peers. To address this problem with proximity-aware unstructured overlay networks, I propose a technique for constructing unstructured overlay networks that simultaneously achieve short path length and reflect the proximity of peers.

To evaluate my proposed approach, I apply it the approach to two existing proximity-aware methods: Localiser [17] and mOverlay [30]. By comparing my method with these two original methods and other methods through simulations, I quantitatively demonstrate the benefits of my approach.

1.2.2 Constructing Balanced and Proximity-Aware Skip Graphs

For its second contribution, this dissertation proposes a new method for constructing skip graphs with low search delay. A skip graph is a structured overlay network consisting of numerous hierarchically organised lists.

In the original skip graph construction algorithm, the overlay topology is determined by the randomly generated *membership vector* of the peers. Thus, the construction does not consider the proximity of peers, which may result in communication links with very large search delay. In addition, because of the random nature of the construction, peers are not necessarily distributed into different lists uniformly, resulting in topological imbalance and increasing path length. This may lead to search inefficiency.

My proposed algorithm is executed by a new peer when it joins the network. The algorithm determines the lists to which each peer belongs by considering the proximity to other peers and the topological balance. In addition, the algorithm is repeatedly executed by each existing peer, allowing dynamic network reconstruction.

The proposed algorithm can also be modified for incorporation into two existing skip graph variants. By comparing the graphs obtained by my approach with the original graphs, I quantitatively demonstrate the benefits of my approach.

1.2.3 Overview of Dissertation

This dissertation is organised as follows. Chapter 2 describes the properties of unstructured and structured overlay networks. Chapter 3 entitled 'Constructing Unstructured Overlay Networks with Low Link Delay and Short Paths', describes

the dissertation's first contribution. This chapter describes a new approach for reducing search delay and path length. The application of the approach to the existing methods for unstructured overlays is also presented. In Chapter 4, entitled 'Constructing Balanced and Proximity-Aware Skip Graphs', the second contribution is described. First, this chapter describes the original skip graph and its limitations. Next, it presents a reconstruction algorithm and its incorporation into construction methods for a skip graph and its variants. Chapter 5 summarises this dissertation and discusses future work.

Chapter 2

Overlay Networks

An overlay network is a virtual network independent of the physical network. In a P2P system, peers typically communicate with each other through this network. Figure 2.1 shows an example of physical and overlay networks. The lower network is a physical network and the upper one is an overlay network. There are two types of overlay networks: the centralised and distributed.

A centralised overlay network, which is used in server/client systems, consists of one server and some peers called clients. Each client sets only one link to the server. Figure 2.2 shows an example of this type. In this overlay, the server manages all the peers in the system. Therefore, searching data and maintaining network connections are easy. However, because the entire load is concentrated on the server, the centralised overlay does not scale. Moreover, a server failure immediately causes the system to fail. On the other hand, in a distributed overlay network, which is used in P2P systems, peers are equally privileged. Figure 2.3 shows an example of such a distributed overlay network. In this overlay, a peer's number of links increases only slowly with the number of peers. Therefore, the

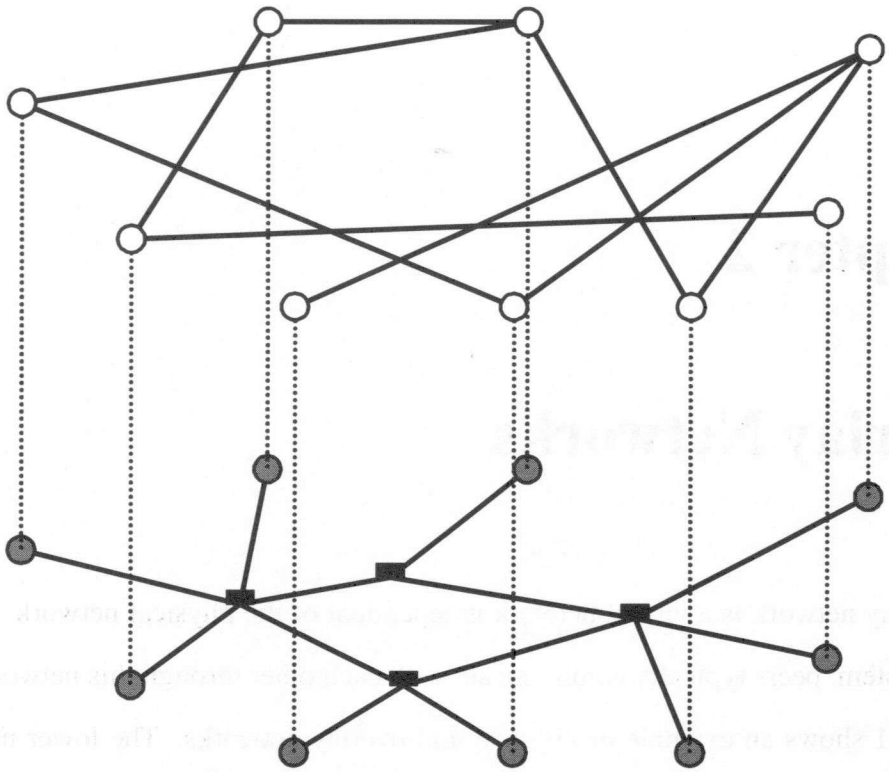

Figure 2.1: Physical Network(lower) and Overlay Network(upper)

required cost per peer to maintain the overlay is lower than that for a centralised overlay. However, because there is no peer that manages the entire network, peers cannot search for data straightforwardly and must periodically communicate with other peers to keep the network connected. In this dissertation, I focus on distributed overlay networks.

The terms and symbols used in this dissertation are summarised as follows:

- I call a network constructed from routers and cables a *physical network*, and a virtual network constructed from end peers (peers for short) connected to the physical network an *overlay network*.

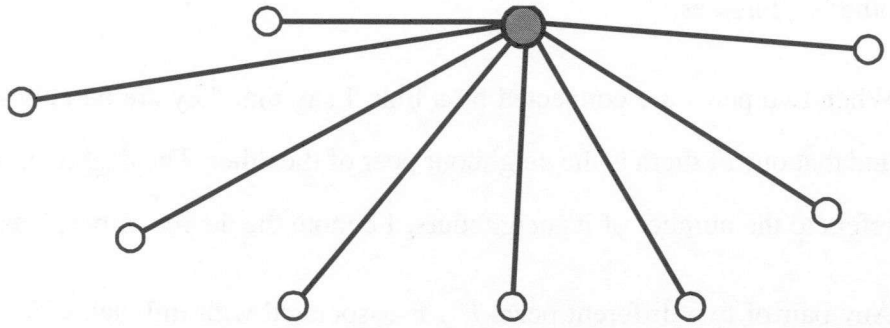

Figure 2.2: Centralised Overlay Network

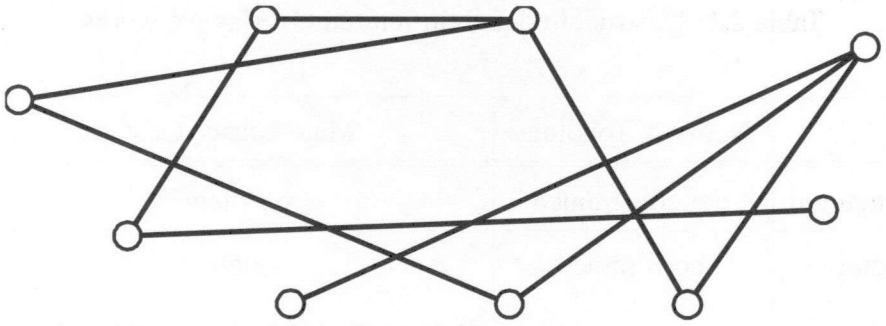

Figure 2.3: Distributed Overlay Network

- In an overlay network, a link connects two peers. In reality, a link could be a TCP connection or could simply indicate that the two peers know each other's addresses.

- When two peers are connected by a link, I say that they are neighbouring, and that one of them is the neighbour peer of the other. The degree of a peer refers to the number of its neighbours. I denote the degree of peer i as d_i.

- Any pair of two different peers i, j is associated with link delay $d(i,j) > 0$, which is independent of the topology of the overlay. Intuitively, $d(i,j)$ represents the delay time required for direct message transfer from i to j. I assume that $d(i,j) = d(j,i)$ for any peer i, j.

Depending on whether there is a stringent network topology constraint, distributed overlay networks are classified into two types: unstructured and structured. Table 2.1 compares these two types.

Table 2.1: Unstructured and Structured Overlay Networks

	Network Topology	Maintenance Cost	Search
Unstructured	not constrained	very low	unreliable
Structured	constrained	high (maintenance messages are required)	reliable

In the following sections, I describe unstructured and structured overlay networks in detail.

2.1 Unstructured Overlay Networks

In unstructured overlay networks, peers need not satisfy any topological constraint. Therefore, in this type, peers can easily maintain a graph topology, and hence, it is suitable for systems with high churn [7, 21]. Several unstructured overlay networks have been proposed [4, 27], however, these have some problems.

The first is the proximity problem. When peers choose their neighbours randomly, some links may have large link delay. If a peer communicates with another peer through a path including such links, this communication requires large delay. Therefore, peers should choose their neighbours with low link delay. However, this resolution increases path length.

Network partitioning occurs with high probability in a graph with a high clustering coefficient. Unstructured overlays are usually fragile against network partitioning, because peers can perceive the status of only a small portion of the entire network.

2.2 Structured Overlay Networks

Structured overlay networks are used for reliable searching. Peers require more information and communication to maintain the graph topology than in unstructured overlay networks. On the other hand, a structured overlay guarantees that a peer can find data if they exist in the network.

In structured overlay networks, the network partitioning problem is not fatal. However, the proximity problem is still important. The chief goal of structured overlay networks is to search for a key. Therefore, search delay significantly affects the performance of such an overlay network.

The most well-known structured overlay network is based on the dynamic hash table (DHT). Many methods are based on DHT [19, 22, 24, 31]; however, they are not appropriate for range searching.

In this dissertation, I focus on skip graphs, one of the most famous structured overlays. In a skip graph, peers can execute range searching easily.

Chapter 3

Constructing Unstructured Overlay Networks with Low Link Delay and Short Paths

3.1 Introduction

One of the most important properties that an overlay should exhibit is short path length. Path length refers to the number of overlay links in a path between two peers. Clearly, path length should be short in order to reduce the number of relay peers that are required to forward a message and thus to achieve efficient communication.

Geographical proximity among peers is also an important feature to consider. The proximity between two peers is usually expressed by the distance between them in the physical network. If no care is taken to reflect the proximity during overlay construction, considerable redundant traffic is produced, resulting in in-

efficient communication and performance degradation of P2P applications. This chapter addresses these two issues in overlay construction.

To construct a proximity-aware overlay network, the existing approaches typically establish overlay links between two peers that are nearby in the physical network [17, 30]. By shortening all links, the average search delay between a pair of peers can be reduced. This approach works well for structured overlay networks, because path length is usually bounded by a small value determined by the number of peers [2, 9, 22, 19, 24]. However, for unstructured overlay networks, this approach increases the path length between two physically distant peers.

To address this problem with proximity-aware unstructured overlay networks, I propose an approach for constructing overlay networks that simultaneously have short path lengths and reflect peer proximity.

My proposed method modifies the existing conventional approach. To reflect geographical proximity, the conventional approach uses short links between two physically nearby peers. In addition to these short links, my method employs some long links that connect two long-distance peers in the physical network. These long links significantly reduce the path length between distant peers, thus solving the problem of the existing approach. Although using a long link means large search delay, I will show later that this does not reduce communication efficiency on average, because most links are short ones connecting nearby peers.

The concept of using long links is not new. For example, the similarity of my approach to the well-known Watts–Strogatz model for small-world networks is obvious. However, the proposed method has several features that distinguish it from previous studies: (1) Long links are associated with a certain distance and are established such that the difference between that value and the actual

distance is minimised. In contrast, the existing methods typically use random links. (2) The number of long links is controlled by a pre-determined parameter. The parameter determines the ratio of long links to all links. This is in contrast to, for example, the GoCast protocol [25], in which a peer has basically one random link. (3) My proposed method can be incorporated into many existing proximity-aware overlay construction methods. The first two features allow more control over the overlay topology than the existing methods. By comparing my method with GoCast through simulations, I will quantitatively demonstrate this benefit.

3.2 Related Work

Several methods are used to construct proximity-aware overlay networks. The Localiser [17] algorithm iteratively changes link connections to reflect the proximity of neighbour peers. In this algorithm, physically close peer pairs are more likely to have a mutual link. This algorithm is robust to churn, since it allows continuous topological changes. Localiser also has a mechanism that uniformly distributes peer degree while maintaining the same number of links. This mechanism provides high resilience to peer failure.

The mOverlay algorithm [30] organises a proximity-aware overlay network in a two-level hierarchy. In this structure, peers compose groups and the groups compose a network. The peers in a group are linked to each other; therefore, the overlay is very resilient to peer failure.

The LTM algorithm proposed in [15] also considers proximity in constructing an overlay network. In this algorithm, each peer repeatedly cuts high-costs links and creates connections with nearby peers.

My proposed method can be incorporated into the existing proximity-aware methods. In this chapter, I extend the Localiser and mOverlay algorithms by adopting the proposed method.

Some methods go one step further; they consider not only proximity but also other properties. GoCast [25] is one such method. In GoCast, most peers have exactly one random link. All other links are chosen on the basis of proximity.

The topology-aware gossip overlay [12, 13] uses an approach similar to GoCast. In this overlay, a peer maintains two lists of links, one containing links to current neighbour peers and the other containing those to random peers. The former link list contains some random links and some short links. These links are used for normal communications. The links in the latter list are used as a fallback when all neighbour peers fail.

In the Foreseer architecture [5], each peer uses both proximity-aware links and friend links for improving search efficiency. Friend links are based on the relationships between the attributes of peers. However, I will not consider the attribute of peers in this dissertation.

These methods construct proximity-aware overlay networks with small path lengths. From these methods, I select the GoCast method to compare with my proposed method.

3.3 Performance Measures

The important properties in an overlay network depend on the requirements of a P2P system. For example, in a file-sharing system such as gnutella [21], a peer must have to be limited number of links in order to avoid concentration of mes-

sages on a few peers, because the overlay network consists of personal computers that do not have high performance. On the other hand, in a system such as the P2P earthquake announcement system [1], the communication delay from one peer to others is the most important property.

In this chapter, I consider systems in which each peer keeps data and provides it to other peers in response to their requests. In addition, I assume the following conditions.

- Peers cannot know which peer has the target data. Therefore, a peer must search for the data by a flooding or a gossip method.
- When a peer finds the target data, it receives the data directly from the peer that has them. To maintain data consistency, peers do not keep a copy of data and a single data item is not provided by more than one peer.
- All peers frequently execute search operations. The cost of processing a search query is not negligible.
- Because a search query consists of only the target data ID and the address of the initiating peer, the size of the search query is very small. Moreover, I do not consider the bandwidth of the network.
- The number of links per peer should be small. For example, in the gnutella network [21], most peers have only 20 or less links.

In this chapter, I use the following four performance measures to evaluate overlay network topologies.

- **Path length:** The length of a path in an overlay is the number of links in the path. Path length for two peers is defined as the path length for the

shortest path between them. In this chapter, I use the average path length for all peer pairs as a performance measure for an overlay network.

- **Search Delay:** Search delay of a path is the sum of the delay of the links constituting the path. I define the search delay between two peers as that of the shortest path between them.

- **Clustering Coefficient:** This measure quantifies how close a given peer and its neighbours are to being a clique [26]. Note that a total of $d_i * (d_i - 1)$ peer pairs can be selected from the neighbours of a peer i. The clustering coefficient of a peer i is defined as the ratio of neighbouring peer pairs in those $d_i * (d_i - 1)$ peer pairs. The clustering coefficient of the entire network is the average of the clustering coefficients for all peers. The clustering coefficient should be small, since areas of the overlay having a high clustering coefficient are easily disconnected by peer failure.

- **Reachability:** Given a set of failed peers, I define reachability as the ratio of peers in the largest fragment of the network to peers that have not crashed. For example, assume that the total number of peers is $10,000$ and the failure ratio is 0.2. Then the number of operational peers is $8,000$. Now, suppose that the network is partitioned into fragments because of peer failure, and that 400 operational peers cannot be reached from the largest fragment of the network. In this case, reachability is $\frac{7600}{8000} = 0.95$.

3.4 Existing Proximity-Aware Methods

3.4.1 Localiser

Localiser [17] is a fully decentralised algorithm that iteratively reshapes the topology of an overlay network. In this algorithm, each peer repeatedly replaces its own links with shorter ones. As a result, the overlay network gradually becomes close to the physical network.

The following steps show how a peer, say i, replaces its links; w and T are parameters. (Figure 3.1 shows these steps schematically).

1. Randomly choose two of its neighbours, peers j and k, and measure link delays $d(i,j)$ and $d(i,k)$.

2. Send messages to peers j and k, which return d_j and d_k respectively. In addition, peer j sends back its estimate of $d(j,k)$.

3. Evaluate locally the cost of replacing link (i,j) with link (j,k). The cost is defined as $\Delta E = 2w(d_k - d_i + 1) + d(j,k) - d(i,j)$.

4. Replace the links with probability $p = \min(1, (e^{-\Delta E/T} \frac{d_i(d_i-1)}{d_k(d_k+1)}))$.

3.4.2 mOverlay

The mOverlay algorithm [30] constructs a two-level hierarchical network. The top level consists of groups of peers and the bottom level consists of peers within each group. Figure 3.2 represents an mOverlay network schematically.

Each group consists of nearby peers that have links with each other. At the top level, a group has some meta links to nearby groups. A meta link is implemented

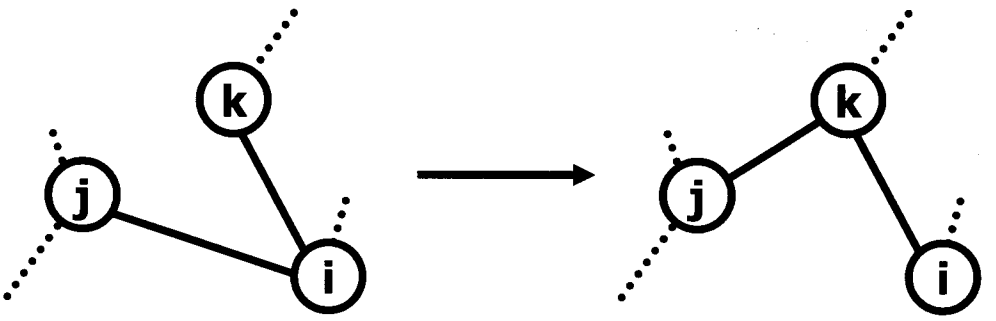

Figure 3.1: Localiser Algorithm

by a set of unidirectional links. If two groups are connected by a meta link, all peers in either group have links to H peers in the other group, where H is a design parameter. The two-level hierarchical structure can thus allow efficient communication.

When a peer joins the network, it first searches for a nearby group. If a group that meets a criterion is found, the peer joins that group. Otherwise, a new group is created and the new peer joins the new group.

Creating a new group involves selecting M neighbouring groups, where M is a design parameter. M meta links are added between the new group and these M groups. The selection of the neighbouring groups is conducted by invoking the following procedure M times for finding a nearby group.

1. Consult with a special server called the rendezvous point to obtain the address of an existing peer called a *boot host.*

2. Measure the distance C_G to the group, say G, of the boot host by measuring the average search delay to all peers in the group.

3. Let $\mathcal{G}$ be the set of the neighbouring groups of G. Measure the distance to

each group in $\mathcal{G}$.

4. If a stop criterion is met[1], go to Step 5. Otherwise, set G to a group in $\mathcal{G}$ such that $C_G = \min_{G' \in \mathcal{G}}\{C_{G'}\}$ and go to Step 3.

5. Let $\mathcal{G}'$ be the set of all groups whose distance has been measured. Select a group G from $\mathcal{G}'$ such that $C_G = \min_{G' \in \mathcal{G}'}\{C_{G'}\}$.

The group selected by this procedure varies depending on the boot host. If the same group has been selected more than once, the total number of nearby groups obtained becomes less than M. In that case, new groups are selected from the neighbouring groups of these selected groups such that a total of M groups are eventually selected.

3.4.3 Problem with Traditional Proximity-Aware Methods

In overlay networks constructed by these proximity-aware methods, physically distant peers tend to have only long paths. This can cause significant messaging delay because message forwarding entails non-negligible overhead [6, 14]. Broadcasting is also affected by this problem; in P2P applications, broadcast is often used for many purposes, for example, to search for a peer that has a required resource. Each broadcast message is associated with a maximum number of hops that it can go through in order to prevent it from travelling in the network forever. This number is usually called *time-to-live* (TTL). Because of this limitation, the message can be discarded before reaching the target peer if the path length between the peer initiating a broadcast and the target peer is greater than TTL.

[1] In the simulations I conducted, I stopped the search when the next G would be the group that has already been selected as G.

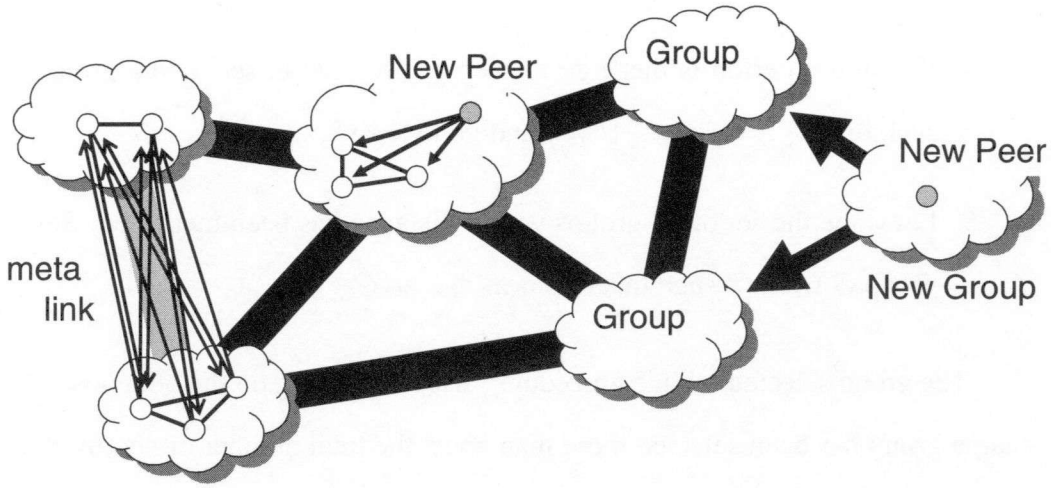

Figure 3.2: mOverlay Network

3.4.4 GoCast

GoCast [25] is a method for constructing an overlay network with small path length and low search delay. This method employs two types of links: short links and random links. Figure 3.3 shows an example of the GoCast overlay network. Most peers have exactly one random link. All other links are chosen on the basis of proximity.

1. When a peer joins the overlay network, it selects some physically nearby peers. It also selects another peer at random. Then, the peer establishes links between these selected peers.

2. If each peer recognises that it has more links than it should have, it cuts some high-cost links such that the change in its neighbours' peer degree are minimised (Figure 3.3). On the other hand, if the peer has lesser links than it

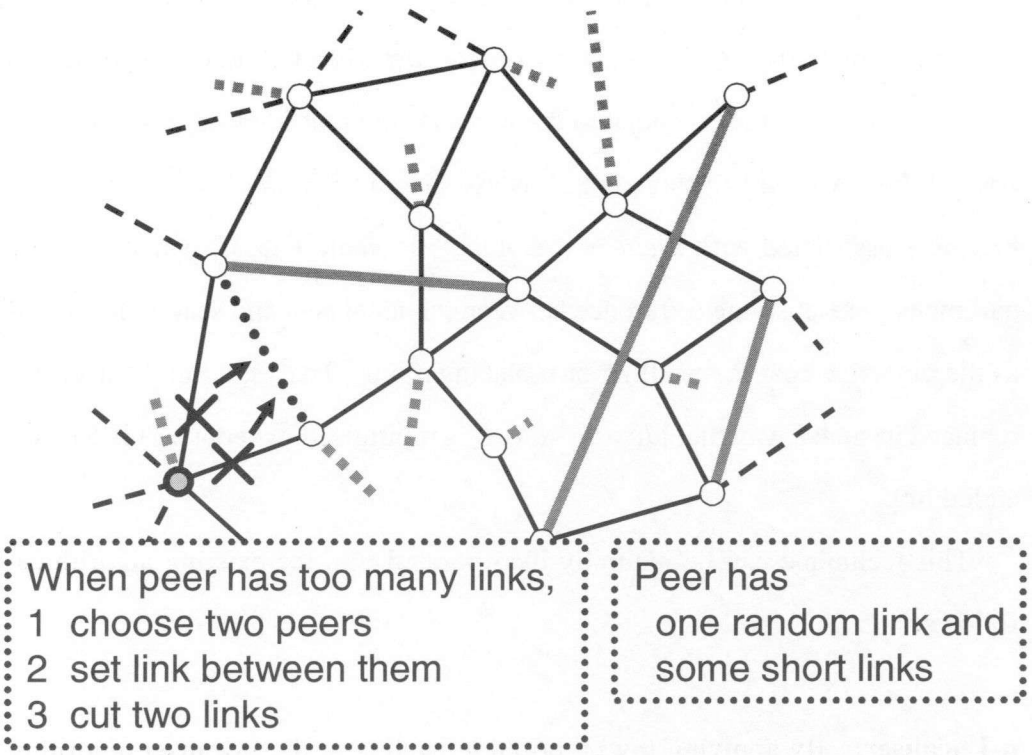

Figure 3.3: GoCast Network

should have, it selects peers as in the joining step and creates links to them.

3.5 My Proposed Technique

The aforementioned problem occurs because previous proximity-aware methods install links only between geographically nearby peers. My technique alleviates this problem by introducing *long links* in the overlay network. The technique consists of two components: long link selection and objective cost assignment.

A long link is selected whenever a new link is added to the network for the first time. The new link is selected as a long link with probability P; otherwise, it

is a short link. P is a design parameter that must be decided a priori.

A new link is also associated with its *ideal cost* when it is added to the network. Assume that a new link is added to the network and that $ideal$ represents its ideal cost. If the new link is a short link, it is associated with $ideal = 0$. If it is a long link, it is associated with $ideal = GOAL \geq 0$, where $GOAL$ is another design parameter. The absolute difference between the ideal cost and search delay works as the objective cost in installing or replacing a link. That is, a link is installed or replaced in such a way that $|d(i,j) - ideal|$ is minimised, where (i,j) is the newly added link.

This technique can be naturally incorporated into the existing algorithms as follows.

p-Localiser: By applying my proposed technique to the Localiser algorithm, I have the following new algorithm, which differs from the original in that Step 3 uses a different cost function. I call the algorithm *p-Localiser*. Here, peer i initiates the algorithm. Whether each link is long or short is determined when the original network is built.

1. Randomly choose two neighbours, say peers j and k, and measure link delays $d(i,j)$ and $d(i,k)$.

2. Send messages to peers j and k, which return d_j and d_k respectively. In addition, peer j returns its estimate of $d(j,k)$.

3. Evaluate locally the cost of replacing link (i,j) with link (j,k). To reflect the ideal cost, the cost is now defined as $\Delta E = 2w(d_k - d_i + 1) + |d(j,k) - ideal| - |d(i,j) - ideal|$, where $ideal$ is the ideal cost of the link

currently connecting i and j.

4. Replace the link with probability $p = \min(1, (e^{-\Delta E/T} \frac{d_i(d_i-1)}{d_k(d_k+1)}))$. If replacement occurs, the new link (j, k) inherits the type (i.e. short or long) of the removed link (i, j).

Preliminary results for the case $GOAL = \infty$ are given in [16].

p-mOverlay: My proposed technique can be naturally incorporated in the process of installing meta links in mOverlay. I refer to this new version of mOverlay as *p-mOverlay*. In the original mOverlay, when a new group is created, it selects neighbouring groups such that the distance to them is minimised. In contrast, the proposed technique modifies this selection process by considering the ideal cost. As a result, meta links are selected as long links with probability P and are added to groups whose distance from the new group is close to the ideal cost. I let $ideal$ denote the ideal cost of a meta link.

Incorporation of the proposed technique amounts to a slight change in the procedure for finding a nearby group. In particular, Steps 4 and 5 are modified as follows:

4. If a stop criterion is met, go to Step 5. Otherwise, set G to a group in $\mathcal{G}$ such that $C_G = \min_{G' \in \mathcal{G}}\{|C_{G'} - ideal|\}$ and go to Step 3.

5. Let $\mathcal{G}'$ be the set of all groups whose distance has been measured. Select a group G from $\mathcal{G}'$ such that $C_G = \min_{G' \in \mathcal{G}'}\{|C_{G'} - ideal|\}$.

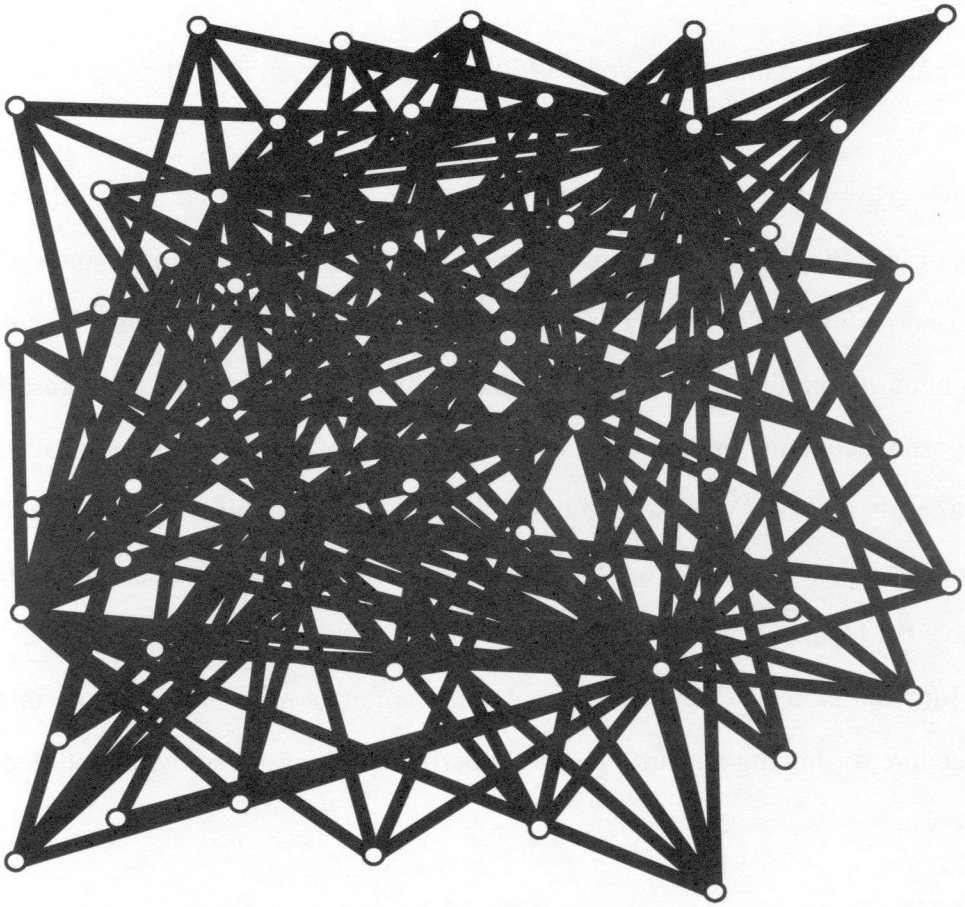

Figure 3.4: Random Overlay Network

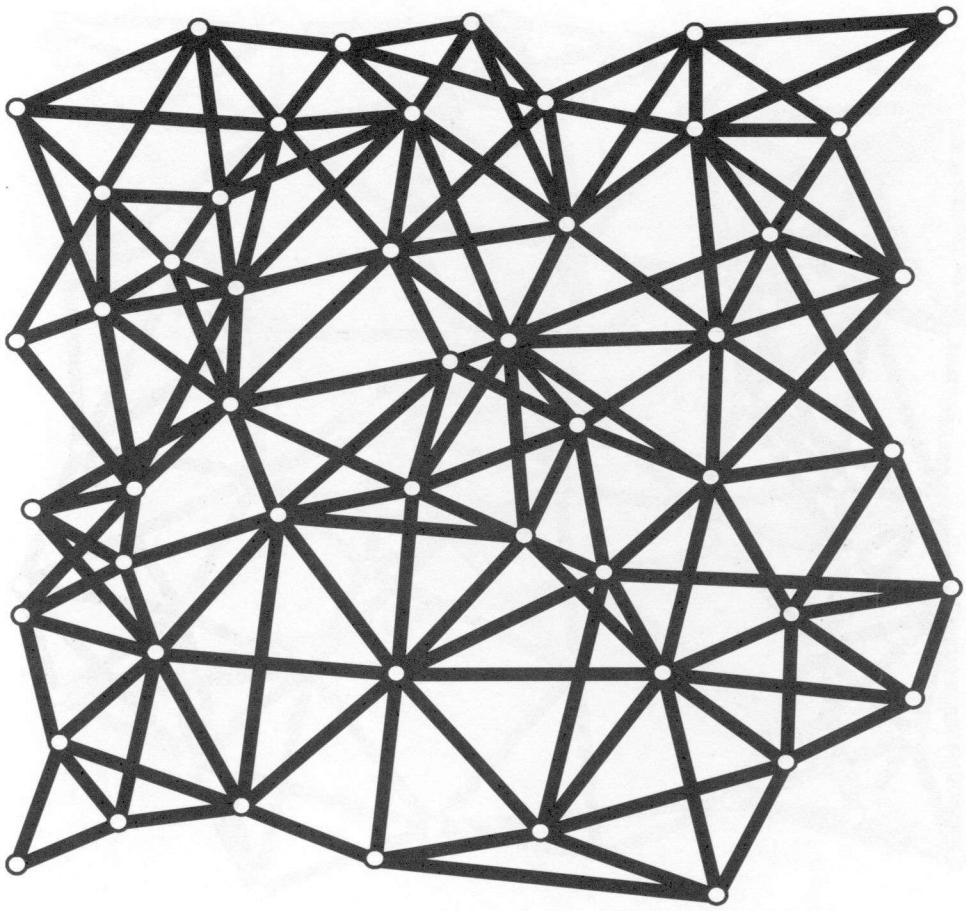

Figure 3.5: Overlay Network Constructed by Localiser

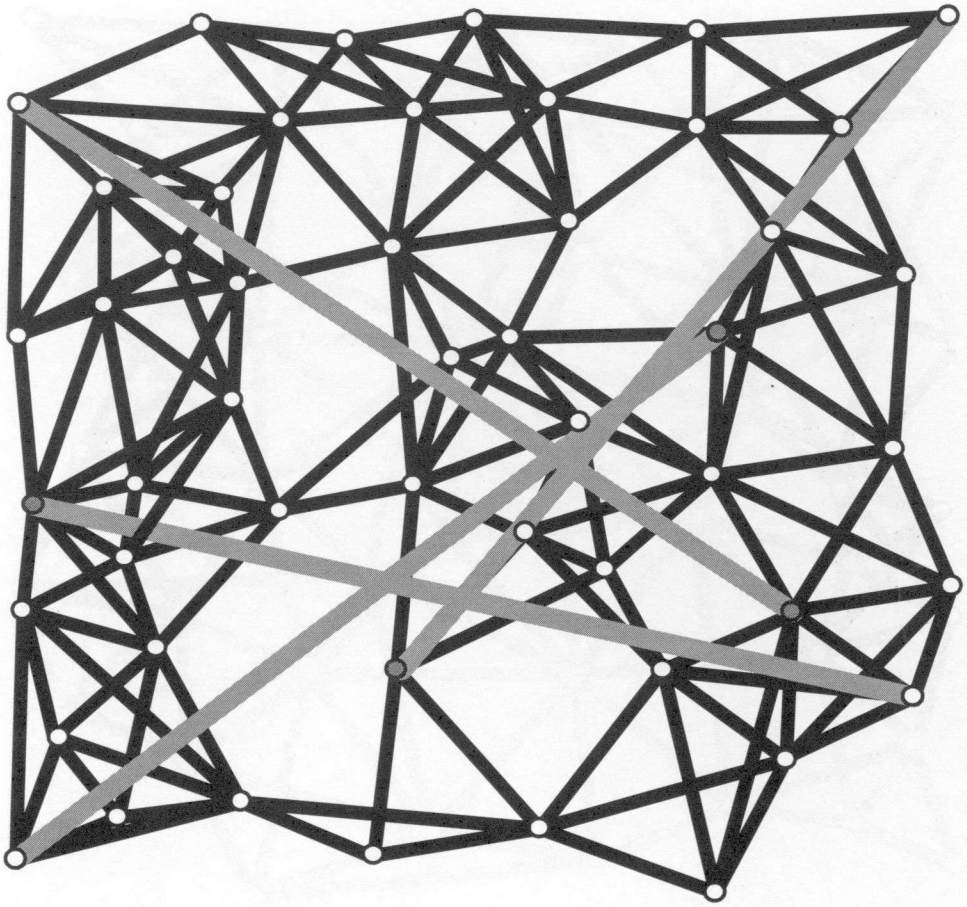

Figure 3.6: Overlay Network Constructed by the Proposed Approach

3.5.1 Illustrative Example

Here, I describe an illustrative example. Figure 3.4 shows a random overlay network with 50 peers. Figures 3.5 and 3.6 show overlay networks obtained from the network in Figure 3.4 by executing the original Localiser algorithm and the p-Localiser algorithm, respectively. The black lines represent short links and the grey lines represent long links. The number of links is the same in Figures 3.4, 3.5 and 3.6.

3.6 Simulation Evaluation

In this section, I present the simulation results with and without my proposed technique for comparison.

3.6.1 Simulation Settings

I used the Georgia Tech transit-stub model [28] to create physical networks. Each physical network is composed of 100 transit domains, each of which has 100 stub domains. Link delays are modelled by simply assigning propagation delay of around 10ms to each physical link between two transit domains and 1ms to other physical links.

The link delay between two peers in the overlay network is the communication delay of the shortest path in the physical network, where the communication delay in the physical network is the sum of the delays of the physical links forming the path.

The transit domains are located in a two-dimensional space. Each is connected

to three other nearby transit domains on average. Each transit domain has 100 routers and each router has one stub domain. Each overlay peer joins one of these stub domains. In these simulations, the average link delay between two peers is about 50ms, whereas the maximum delay is about 140ms.

In Simulations 1 and 2, the average degree of peers is set to 15 in random overlay networks, Localiser, p-Localiser and GoCast. In contrast, the average degree is lowered to 6 in Simulation 3 to clarify the difference among different methods, since networks with a high average degree result in near 100% reachability regardless of the construction method. Note that in my preliminary experiments, I confirmed that a different average degree does not change the qualitative properties of these methods. Unlike these algorithms, the mOverlay algorithm has a two-level hierarchical structure; thus, its design parameters are set differently. I set its design parameters such that in the network created, each group had approximately 10 peers on average and $M = 3$ meta links. Peers have $H = 5$ unidirectional links in a meta link.

The behaviour of the Localiser algorithm was simulated as follows. I first created a random overlay network and applied Localiser and p-Localiser to it. In each instance of the simulation, the links were replaced 1000 times by each peer. The parameters w and T were set to $w = 20$ and $T = 50$. With these values, the degree of almost all peers was maintained from 12 to 18.

3.6.2 Simulation 1: Parameters of Proposed Technique

Simulation 1 was conducted to investigate the effects of the parameter values of my proposed technique. I tested Localiser and mOverlay equipped with my technique. In this simulation, I assumed that there were 10,000 peers in the overlay

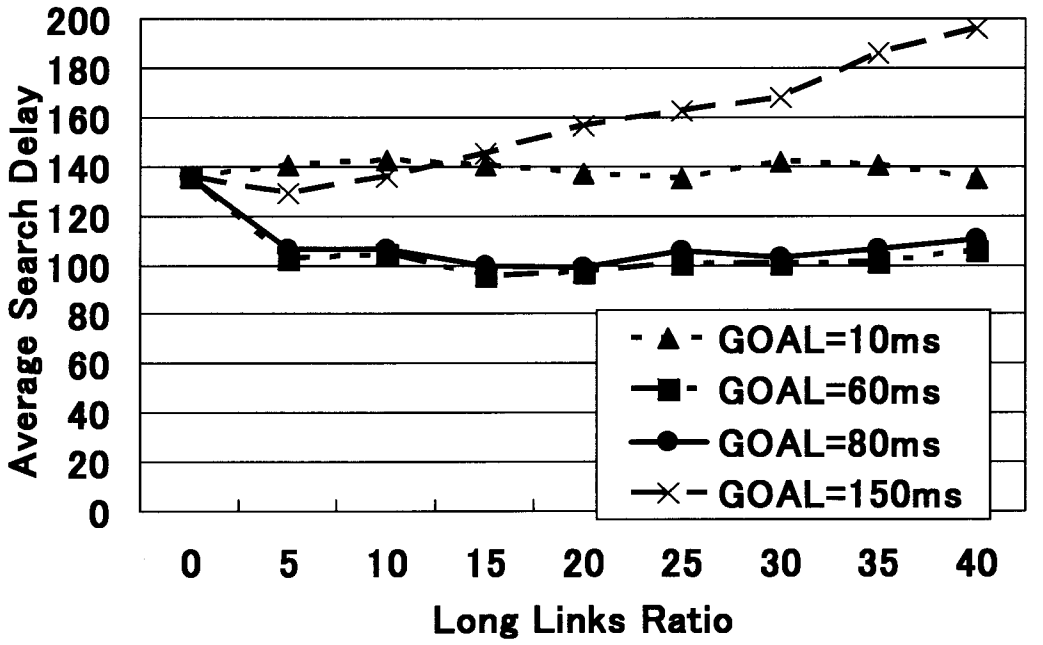

Figure 3.7: Ratio of Long Links and Average Search Delay (p-Localiser)

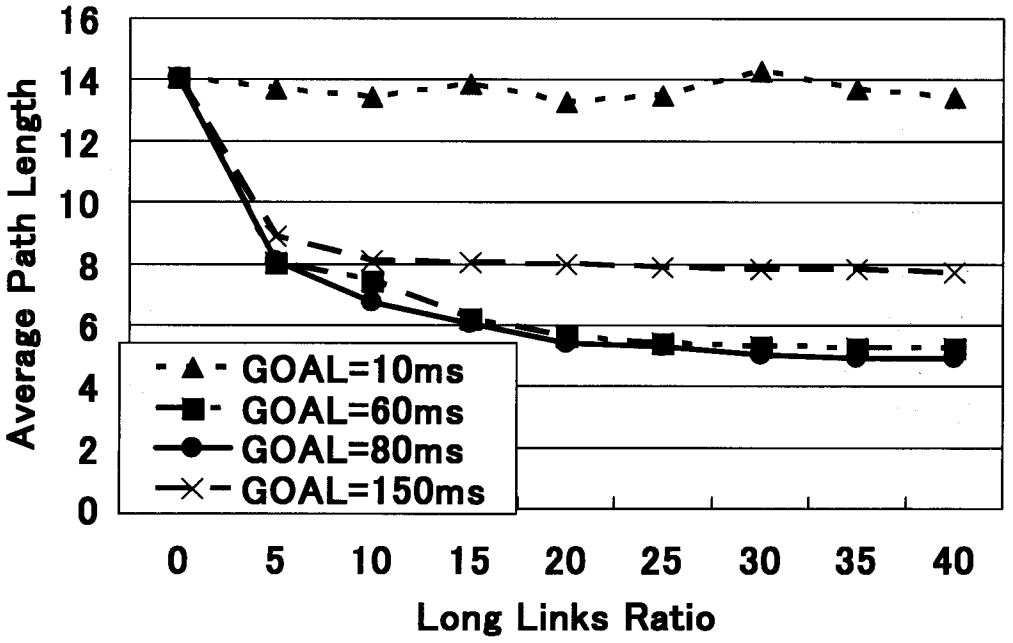

Figure 3.8: Ratio of Long Links and Average Path Length (p-Localiser)

network. Failures were not considered.

I varied $GOAL$, the ideal cost for a long link, from 10 to 150ms, and P, the probability that a new link becomes a long link, from 0% to 40%. Since the link delay between two peers is at most 140ms, when $GOAL$ is 150ms, long links are placed such that their distance is maximised. The original Localiser algorithm is equivalent to p-Localiser if P equals 0.

Figure 3.7 shows the relationship between the ratio of long links P and the average search delay between any two peers. The result for $GOAL = 150\text{ms}$ shows that the average search delay increases rapidly as P increases. At the other extreme, when $GOAL = 10\text{ms}$, the average search delay does not change clearly as P varies. In contrast, when $GOAL = 60\text{ms}$ or 80ms, the average search delay is significantly reduced. Figure 3.8 shows the relationship between P and the average path length. When $GOAL = 80\text{ms}$, path length shows the greatest reduction. However, the reduction is saturated when P exceeds 20%.

From Figures 3.7 and 3.8, I conclude that my proposed technique performs well with $GOAL = 80\text{ms}$ and $P = 20\%$ when applied to Localiser. This is also the case for mOverlay (Figures 3.9 and 3.10). We, therefore, use these values in the following simulations.

The best value of $GOAL$ depends on the communication delay of the physical network. Therefore, the value of $GOAL$ should be determined for each system. However, a rule of thumb can be applied to any system. Simulations show that the best value of $GOAL$ is about half of the largest value of the link delay between two peers. I expect that using half of the largest link delay will yield good results in any system.

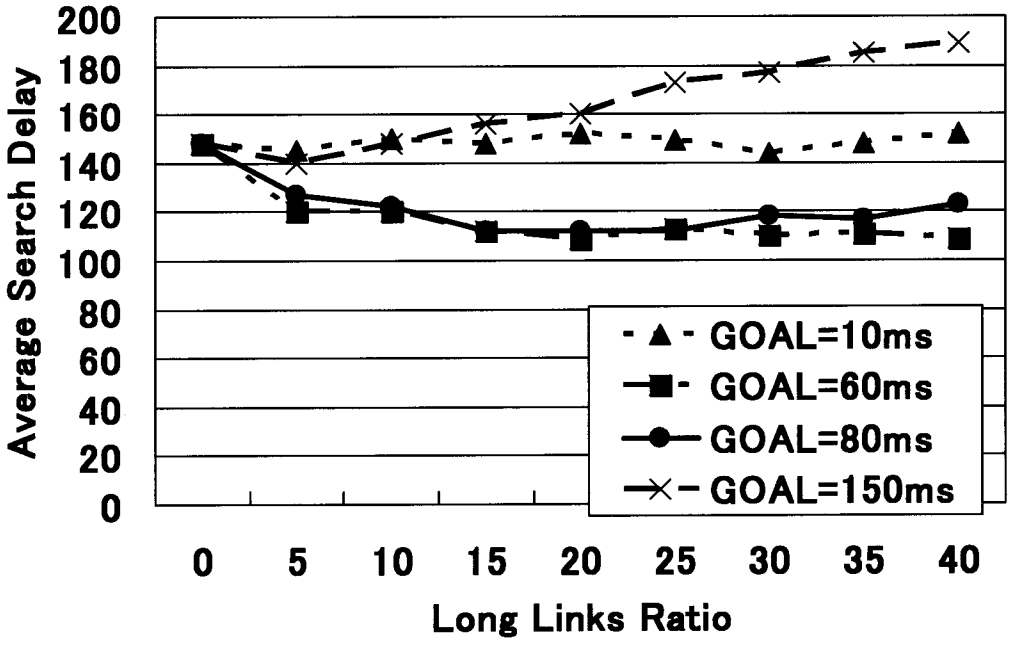

Figure 3.9: Ratio of Long Links and Average Search Delay (p-mOverlay)

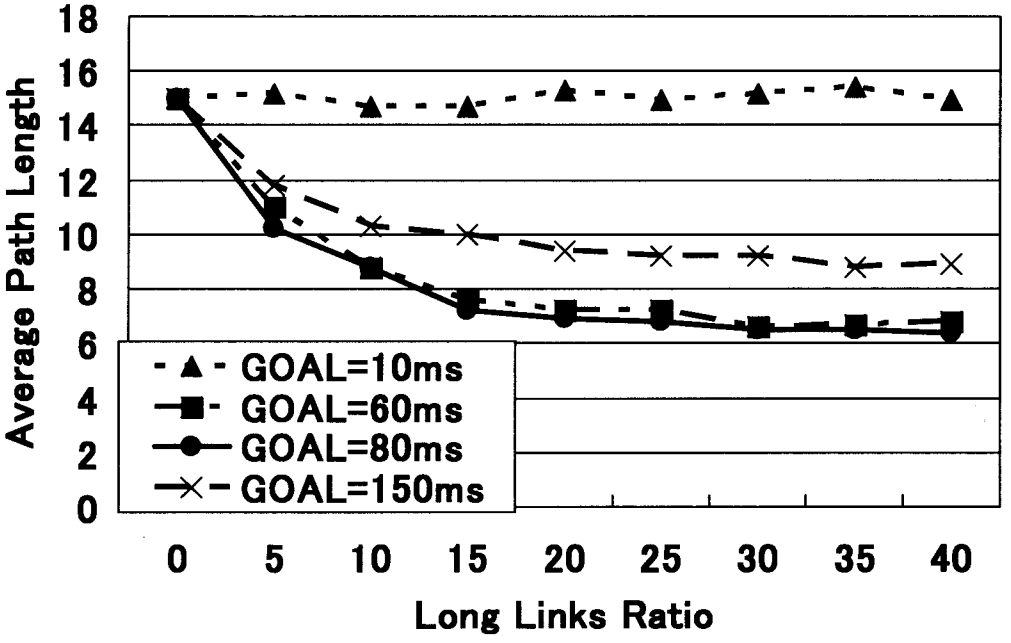

Figure 3.10: Ratio of Long Links and Average Path Length (p-mOverlay)

3.6.3 Simulation 2: Comparison with Existing Methods

In this simulation, I evaluated the performance of Localiser and mOverlay with and without my technique. I measured search delay and path length with varying network size: I varied the number of peers from 2,500 to 20,000. On the basis of the results of Simulation 1, I set $GOAL = 80\text{ms}$ and $P = 20\%$.

Search Delay

Figure 3.11 presents the average search delay for Localiser. This figure compares GoCast, Localiser and p-Localiser, as well as the initial random network to which Localiser was applied. Search delay for p-Localiser is slightly lower than those for GoCast and Localiser. Compared with the initial random network, on the other hand, p-Localiser achieves much lower search delay.

Figure 3.12 shows the result for mOverlay. The benefits of using the proposed technique are much clearer in this case. In the network constructed by mOverlay, distant peers must use meta links between different groups. These meta links are few in number and they connects only nearby groups in the original mOverlay algorithm. As a result, long meta links installed by the proposed technique have a much clearer effect on search delay than in the case of Localiser, which produces flat-structured overlays.

Path Length

Figure 3.13 shows the path length results for Localiser. The results clearly show the advantage of using the proposed technique: with respect to the average path length, p-Localiser achieves significantly lower values than GoCast and Localiser.

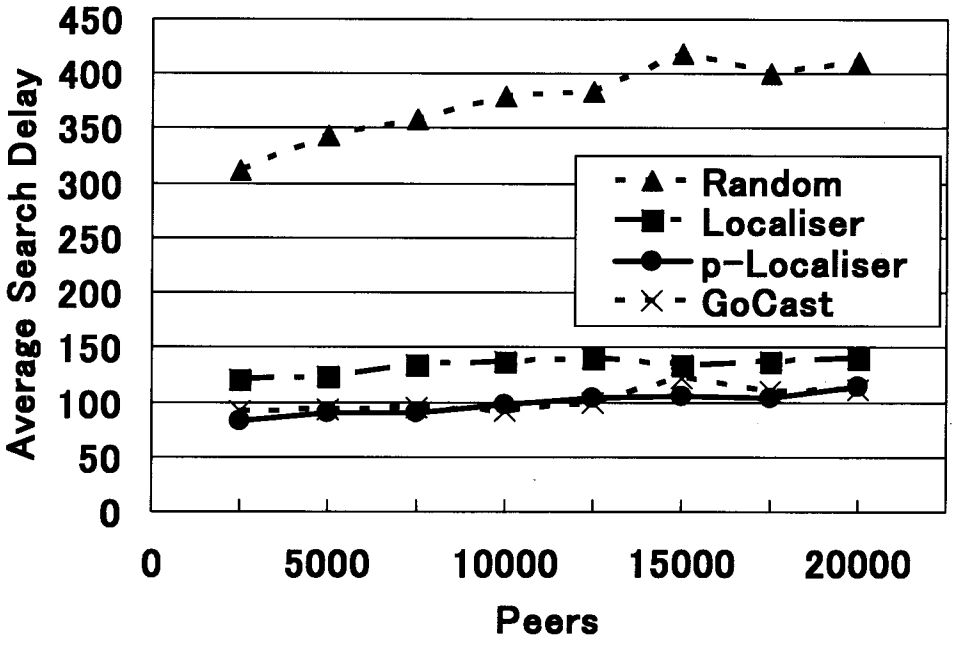

Figure 3.11: Average Search Delay (Random, Localiser, p-Localiser, GoCast)

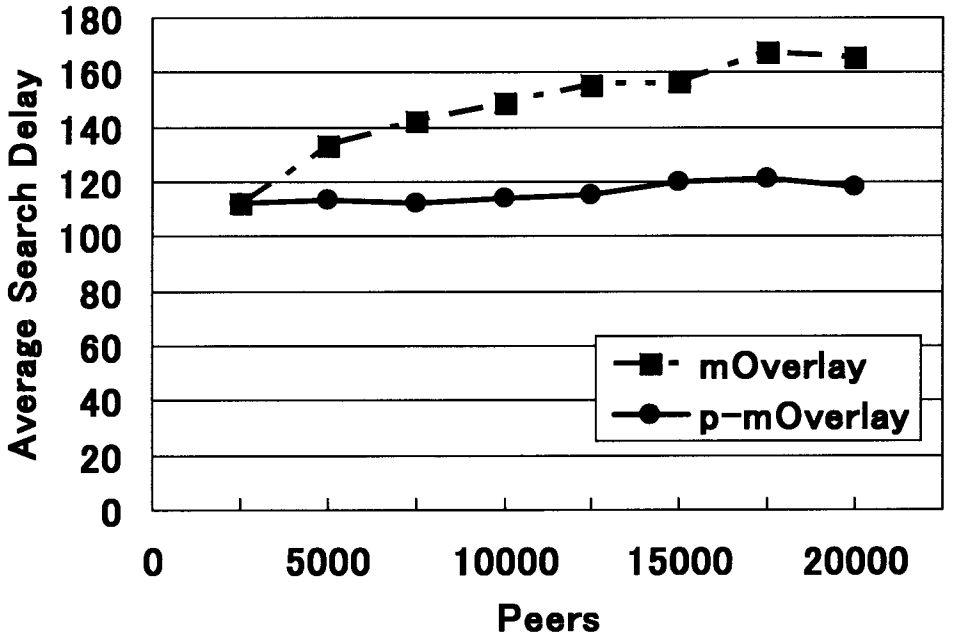

Figure 3.12: Average Search Delay (mOverlay, p-mOverlay)

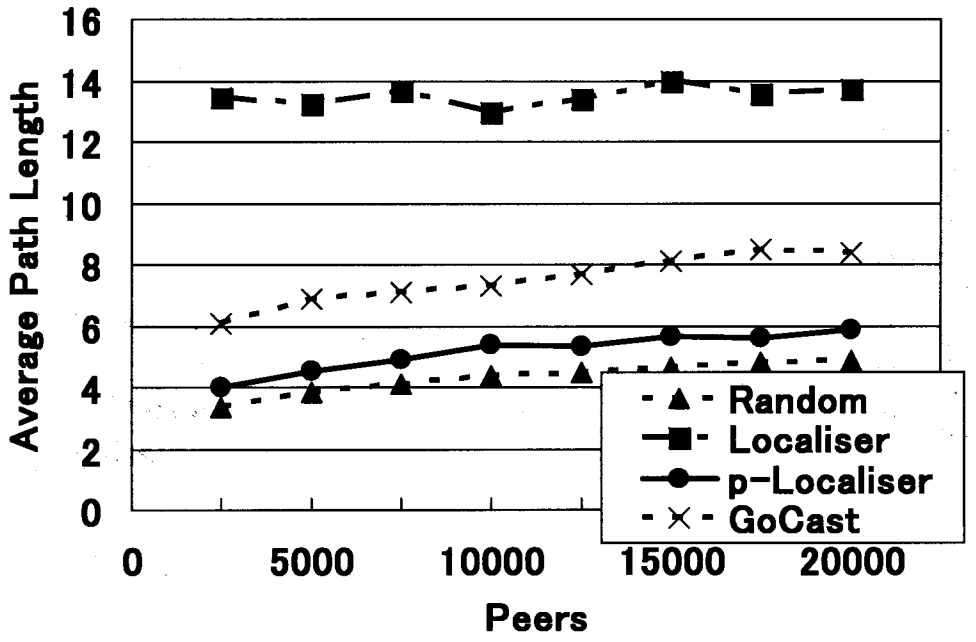

Figure 3.13: Average Path Length (Random, Localiser, p-Localiser, GoCast)

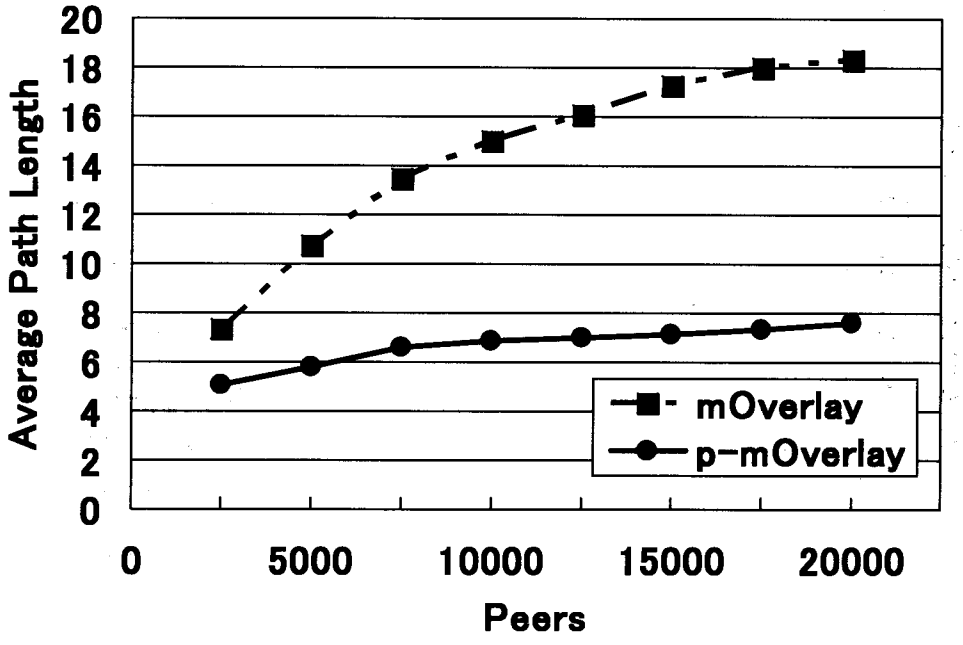

Figure 3.14: Average Path Length (mOverlay, p-mOverlay)

Table 3.1: Clustering Coefficient, Path Length and Search Delay

	Clustering Coefficient	Path Length	Search Delay
Random	0.0023	4.3	379
Localiser	0.59	12.9	136
p-Localiser	0.36	5.4	98
GoCast	0.40	7.3	92

As shown in Figure 3.14, my technique also considerably reduces path length for mOverlay. This reduction in path length significantly enhances the reachability of broadcast messages because it prevents expiration of their TTLs.

Moreover, with my proposed technique, path length increases only moderately as the number of peers increases. That is, my technique improves proximity-aware methods in that the constructed network can better scale to network size.

Clustering Coefficient

I compared the clustering coefficients for the original random network, and those using Localiser, p-Localiser and GoCast when the number of peers is $10,000$. Table 3.1 shows the clustering coefficients obtained by these four methods, as well as the average path length and average search delay.

The simulation results agree with the well-known fact that random networks have low average path length and a small clustering coefficient. Networks with a large clustering coefficient and low average path length are called small-world networks [20]. The networks constructed by p-Localiser and GoCast exhibit small worldness, whereas the Localiser network has large path length and thus does not

exhibit this characteristic.

As stated in Section 3.3, a large clustering coefficient implies vulnerability to failure. Although p-Localiser produces networks with relatively high clustering coefficients, the value is significantly smaller than those for Localiser networks. This observation suggests that my method enhances the fault tolerance of the original Localiser algorithm. In the next set of simulations, I show that this is indeed the case.

3.6.4 Simulation 3: Resilience to Peer Failure

In Simulation 3, I evaluate the resilience of the overlay networks to random peer failures. I assume random failures of peers in the overlay network. In P2P applications, peer failures occur not only when peers crash but also when peers join and leave the network. In such applications, users frequently join and leave the network at will. Peers that have left the network cannot be distinguished from those that have crashed. This means that usually a very large fraction of peers fail simultaneously.

Figures 3.15 and 3.16 show the relationship between the ratio of failed peers and reachability. In Figure 3.15, my proposed method exhibits much higher reachability than the Localiser method does. This is explained as follows. In the original Localiser, peers in the constructed network have links to only nearby peers. Thus, nearby peers share most of their neighbours or the neighbours of their neighbours. As a result, the failure of a peer affects many of its neighbours simultaneously, resulting in a high probability of network partitioning. The long links added by the proposed technique decrease this probability, making the network more resilient to random failures. The high reachability that GoCast exhibits is

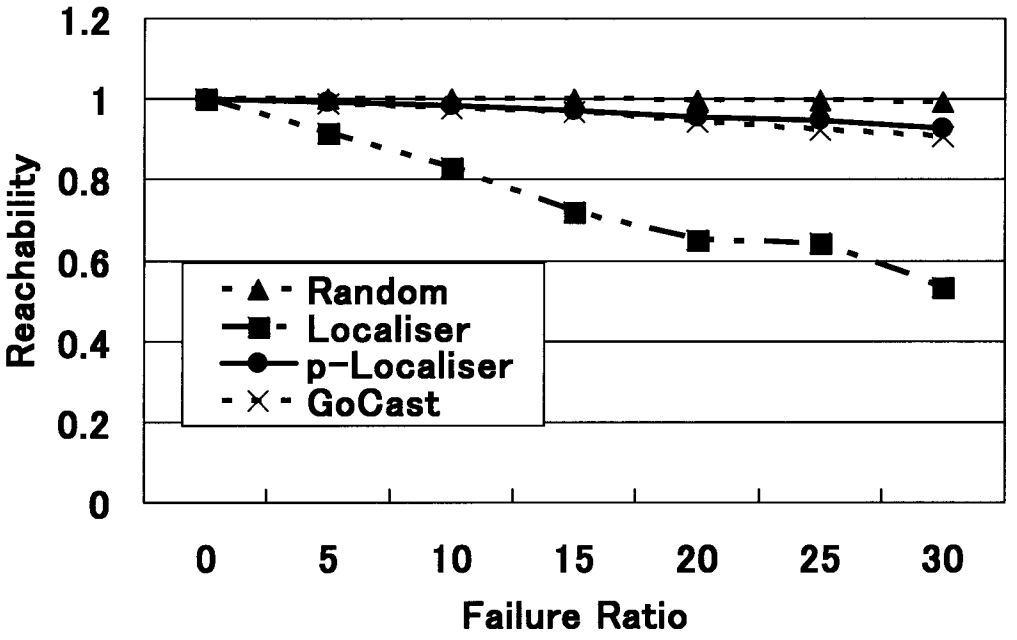

Figure 3.15: Random Failure Rate and Reachability (Random, Localiser, p-Localiser, GoCast)

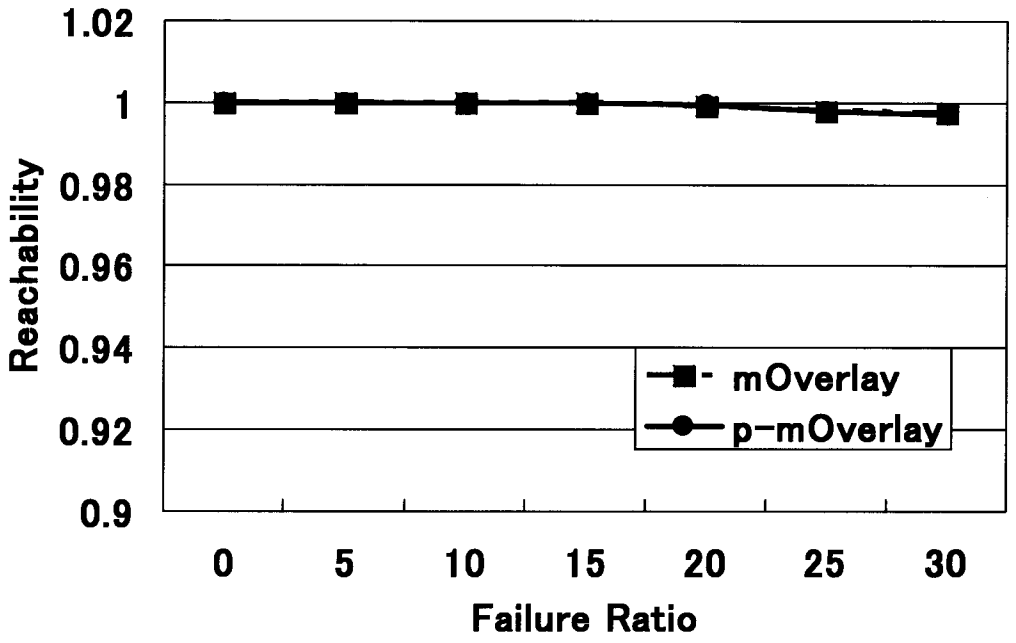

Figure 3.16: Random Failure Rate and Reachability (mOverlay, p-mOverlay)

also explained by the same argument, except that in GoCast high resiliency results from using random links instead of long links.

Figure 3.16 shows the results for mOverlay. mOverlay, by its design, achieves very high resilience to random peer failures. All peers in each group of mOverlay share mutual links, and a meta link between different groups is shared by all pairs of peers between the two groups. Because of this property, network partitioning does not occur unless all peers in a group have failed.

3.7 Discussion

In this chapter, I assumed the following about the overlay network.

- Peers must propagate search queries by flooding or gossip.
- Peers do not cache data.
- Forwarding search queries incurs some cost.
- The size of a search query is sufficiently small compared to the network bandwidth.
- The number of links per peer should be small.

I employ these assumptions to specify the properties of the systems and simplify my argument. However, I expect that my proposed approach can be applied to systems that do not meet these assumptions.

Using caches when receiving a data item is a suitable approach to improve the discovery ratio of the data. If more than one copy of the target data exists in the network, a search query easily finds the target data. However, the cache

also causes difficulties in updating the data. Even if the peer with the original data item updates it and broadcasts the information to all the peers in the network, peers that are detached from the network at the moment of the update may retain the old cache. Therefore, it is hard to update all the cached replicas perfectly. Moreover, when a peer finds a cache of the target data, it cannot judge whether the data are the latest version. If the peer requires the latest version, it must search for the original data. This problem nullifies the advantage of the cache. Therefore, I assumed that peers do not cache data. However, the advantages of my proposed approach do not depend on whether caches exist. Even in a system in which peers use a cache, I believe that my proposed approach reduces both path length and search delay.

The cost of reading and forwarding a search query is generally not high. Recent high-performance machines can process a search query by using a fraction of their processing power. However, when the machine is executing a large process that requires massive machine power, the cost of processing the search query is not negligible. A peer must receive the target data and send them to other peers. Moreover, the peer also executes its own operation on the received data. If too many search queries arrive at the peer while it is processing these operations, it cannot forward these search queries immediately. Therefore, the number of times a search query is forwarded should be as small as possible.

A search query does not carry much information. Basically the information consists of the target data ID and the address of the searching peer. Therefore, it is unlikely that the network bandwidth is consumed by search queries. Of course, transferring large amounts of data can consume the network bandwidth. However, more than one path exists between two peers in an overlay network. Even if one

link in the path temporarily exhibits large delay, the peer can use another path with a small delay to send a search query. In an unstructured overlay network, a search query is spread by flooding or gossip. Therefore, the search query arrives at the peer with the target data through the path with the lowest delay. Consequently, I do not consider limitations on the network bandwidth.

When the number of links per peer, which is denoted as the degree of a peer, increases, peers can spread search queries and finish searching operations more rapidly. However, increasing the degree causes problems. One problem is an increase in management cost. In a system with a large number of peers, it is impossible for a peer to maintain links to all the others. Therefore, the number of links per peer must be fixed. Moreover, security problems can occur in a system with a small number of peers and a large number of links. In P2P systems, a peer can provide whatever data it wants. Therefore, it is possible to provide incorrect data or a computer virus. If each peer has links to all others, these malicious data reach all peers in one broadcast. Therefore, even if it is possible for peers to maintain links to all other peers, the number of links must be restricted.

3.8 Summary

I discussed an approach for constructing overlay networks where peer pairs have a small path length and low search delay. I proposed a technique that installs long links in an overlay network. This technique can solve the problem with the existing proximity-aware overlay construction methods, which provide long paths only to distant peers. By incorporating the proposed technique into two such methods, Localiser and mOverlay, I demonstrated that it can be used in combination with

the existing overlay construction algorithms. Using simulations, I evaluated these algorithms and other overlay construction methods with and without my proposed technique. The result shows that the proposed technique significantly reduces path length. In both Localiser and m-Overlay, a reduction of more than 50% is usually achieved compared to the original algorithms. Even compared to Go-Cast, p-Localiser (the Localiser algorithm incorporating the proposed technique) achieves a substantial reduction of around 30% for various network sizes. Moreover, my proposed technique makes the network more robust against a high peer failure ratio. For example, it doubles the message reachability of the Localiser when the ratio of the failed peers is 30%.

In my evaluation, I considered four performance measures: path length, search delay, clustering coefficient and reachability. Many of them are in a trade-off relationship, and thus, which measure is the most important depends on the characteristics of the system that employs the overlay network. If the system exhibits high churn rates, the clustering coefficient is probably the most important. Path length may be the most important if the size of the messages traversing the overlay is large, because forwarding large messages imposes a high load on relaying peers and links. In many P2P applications, the dominant traffic on overlay paths is that of data query messages, which are small. In that case, search delay should be the most important metric.

Chapter 4

Constructing Balanced and Proximity-Aware Skip Graphs

4.1 Introduction

A *skip graph* [2] is a P2P overlay network topology based on a skip list [18]. In a skip graph, like as in many DHT-based overlays [19, 24, 31], a search operation takes only $O(\log n)$ hops, where n is the number of peers. Moreover, skip graphs have an advantage over DHTs in that they support range searching. This distinguishing feature appears because a skip graph consists of a collection of lists in which peers are lexicographically sorted by their keys.

In this chapter, I propose a (re)construction algorithm for skip graphs to further enhance search performance. In the original construction algorithm, the overlay topology is determined by the randomly generated *membership vector* of the peers. Thus, the construction does not consider the proximity of peers, which may result in communication links with quite large link delay. In addition, because of

the random nature of the construction, peers are not necessarily distributed uniformly into different lists, resulting in topological imbalance. This may lead to search inefficiency.

My proposed algorithm is executed by a new peer when it joins the network. The algorithm determines the lists to which each peer belongs by considering the proximity to other peers and topological balance. In addition, the algorithm is executed repeatedly by each existing peer, allowing dynamic network reconstruction. Simulations are used to demonstrate the usefulness of the proposed approach.

The remainder of this chapter is organised as follows: Section 4.2 provides an overview of a skip graph. Section 4.3 presents the proposed construction algorithm. Section 4.4 provides some analytical results for the skip graph structure obtained by the proposed construction. Recent attention to skip graphs has led to the development of some interesting variants. Section 4.5 shows how the proposed construction can be applied to such variants. Section 4.6 describes the simulation results. Section 4.7 concludes the chapter with a discussion of some future research directions.

4.2 Skip Graphs

A skip graph [2] is a structured overlay network topology. Each peer in a skip graph has two fields: *key* and *membership vector*. Without loss of generality, I assume that a key is a positive integer. Let $m(p)$ denote the membership vector of peer p. The elements of $m(p)$ belong to a finite alphabet set Σ. I denote the size of the alphabet by b; i.e. $|\Sigma| = b$. I consider $m(p)$ as an infinite word over Σ, but in practice, only an $O(\log n)$ length prefix of $m(x)$ is needed on average [2].

A skip graph consists of numerous doubly linked lists, which are structured in multiple levels starting from level 0. In each list, peers are lexicographically sorted by their keys. At level 0, all peers belong to one list. At level 1, all peers are separated into b lists. Similarly, all peers in a list of level i are separated into b lists at level $i + 1$.

The membership vector determines which lists the peer belongs to at each level. Two peers p and q belong to the same list at level i iff $m(p)$ and $m(q)$ have an identical prefix of length i. The maximum level of a peer p is the level at which the list that p belongs to contains only p. The *height* of a skip graph is the maximum of all peers' maximum levels.

Figure 4.1 shows an example of a skip graph with $b = 2$. In the rest of this chapter, I consider $b = 2$, which is the most common case in practice.

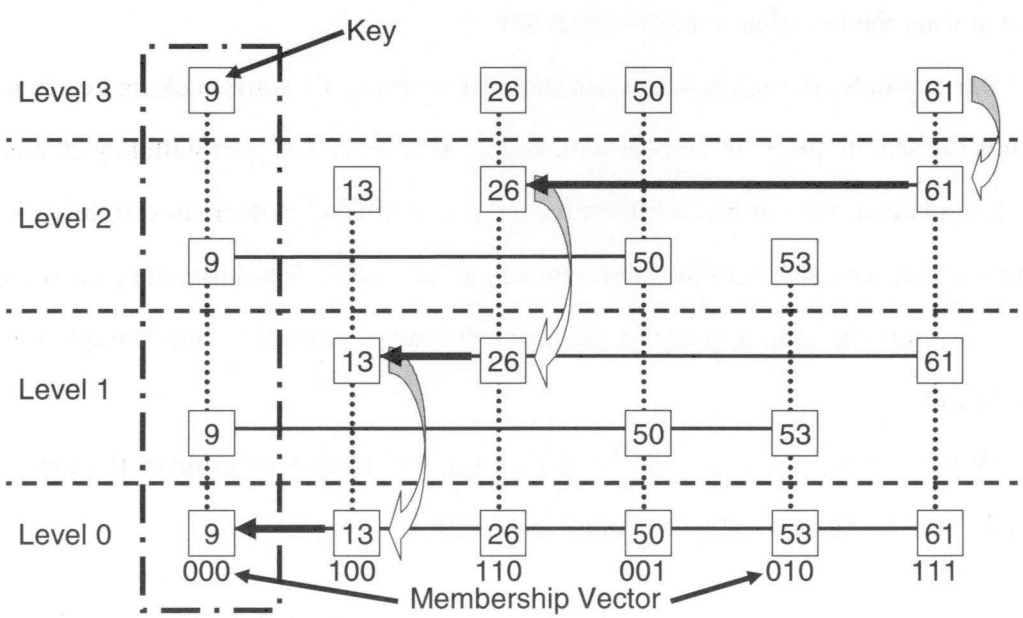

Figure 4.1: Skip Graph

To seek a key, a peer starts the search operation from the maximum level. The initiating peer creates a search query consisting of its address (IP address, in practice) and the target key.

The peer p receiving the search query compares the target key and its own key. If they are the same, the search is completed. Otherwise, the query is passed to an adjacent peer at the current level, or the search level is decreased by one. When the target key is less than the peer's key, the former occurs if the target key is less than the left adjacent peer's key, in which case the query is passed to the left adjacent peer. When the target key is greater than the peer's key, the right peer is selected rather than the left peer. Level decrease occurs if the target key is between the peer's key and the adjacent peer's key. If the level falls below zero, no peer exists that stores the target key, and the peer processing the query is the one storing the key closest to the target key.

For example, in Figure 4.1, when the peer with key 61 starts seeking key 9, it sends the search query to the peer with key 26 at level 2. The peer with key 26 has no left adjacent peer at level 2; therefore, the search level is decreased to level 1, and the peer forwards it to the peer with key 13 at level 1. Similarly, the peer with key 13 sends the search query to the peer with key 9 at level 0, thus completing the search.

When a peer p wants to join the skip graph, it randomly determines the membership vector and executes the following operations.

1. Ask a peer that belongs to the skip graph to search for the key of p. Then find the peer q that has the key nearest to the key of peer p.

2. Send a join message to q and set the links at level 0.

3. Choose the lists to join at levels above level 1 on the basis of the membership vector. Continue this operation until p reaches the maximum level.

When peer p leaves the skip graph, it sends leaving messages to neighbour peers from the maximum level to level 0. In particular,

1. Peer p sends a leaving message to the right and left neighbour peers in the list. This message contains the addresses of these peers.

2. These two peers communicate with each other to construct a link. They send a message to p after setting the new link.

3. Peer p cuts links at the level.

4.3 Balanced and Proximity-Aware Construction

In this section, I propose an algorithm for skip graph construction that considers the proximity of peers and the balance of the topology.

4.3.1 Criterion of Structural Balance

To measure the balance of a skip graph, I introduce the criterion $N_{p,i}$, which is evaluated for a given peer p and a level i. Let $L_{p,i}$ denote the maximum list such that it contains p and is a sub-list of p's level i as well as level $i+1$ lists. $N_{p,i}$ is defined as follows:

$$N_{p,i} = \text{number of peers in the sub-list } L_{p,i}$$

By definition, $L_{p,i}$ contains at least p; thus, $N_{p,i} \geq 1$.

In the extreme case, $N_{p,i} = 1$ holds for any peer p and level i, in which case the skip graph is perfectly balanced. Accordingly, if $N_{p,i}$ is large, the skip graph can be considered imbalanced. In this case, a search query needs to traverse a large number of peers at the same level, thus degrading search performance.

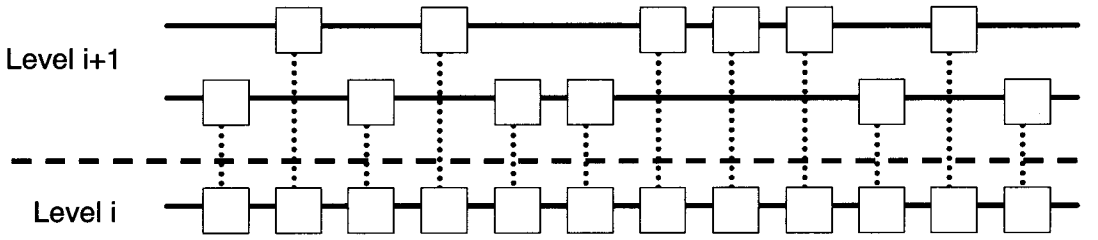

Figure 4.2: Well-Balanced Skip Graph

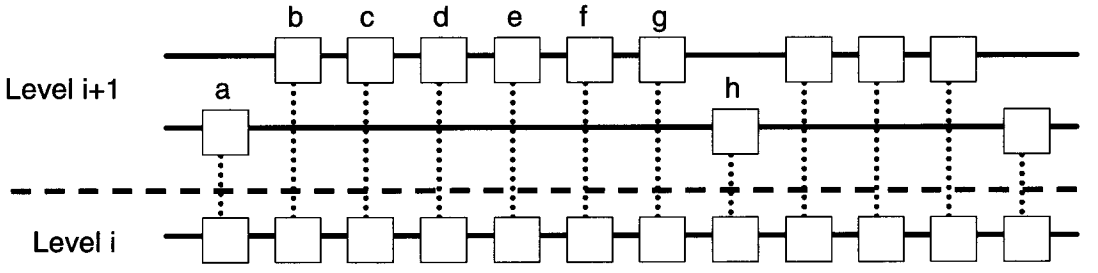

Figure 4.3: Imbalanced Skip Graph

Figures 4.2 and 4.3 show well-balanced and imbalanced skip graphs, respectively. In Figure 4.2, the peers in the list at level i are almost uniformly separated into two lists at level $i + 1$. In Figure 4.3, on the other hand, most of the peers belong to the same list at level $i + 1$. In Figure 4.2, the maximum value of $N_{p,i}$ is 3, whereas it is 6 in Figure 4.3. [In Figure 4.3, $N_{p,i} = 6$ holds for $p =$ b, c, $\cdots$, g, since $L_{p,i} =$ (b, c, $\cdots$, g).]

4.3.2 Topology Reconstruction Algorithm

Here, I describe the proposed algorithm for topology construction. To simplify the presentation I describe this algorithm as a reconstruction algorithm executed by a peer in an already established skip graph network. However, because each newly arriving peer executes the algorithm when it joins the network, this algorithm can be considered as a network construction algorithm. In addition to the newly arriving peers, each existing peer also executes this algorithm for network reconstruction to deal with churns or other changes in the network.

The proposed algorithm consists of three components: (1) reconstruction decision-making based on proximity; (2) reconstruction decision-making based on balance; and (3) reconstruction operation. The algorithm works at each level, from level 0 to the maximum level. Suppose that i denotes the level at which the algorithm works. An outline of the algorithm is depicted in Figure 4.4.

Reconstruction Decision-Making Based on Proximity

This component decides whether the peer should move to the other list. In this component, I use link delay, which is the required communication time between two peers with a link. The steps involved are as follows:

1. Measure the link delay between p and each adjacent peer at level $i + 1$. If no adjacent peer exists (that is, p is at the end of the list), the delay is considered to be zero. Add the delay for the two directions. In the rest of this section, I refer to this sum simply as link delay at level $i + 1$.

2. Estimate link delay at level $i + 1$ assuming that p moves to the other list at this level. This is done by measuring the link delay between p and each of

for $i = 0$ to p's height-1 **do**

Perform decision-making based on proximity.

Let $decision$ be the decision. $\{true$ if reconstruction is decided; $false$ otherwise.$\}$

Let N be the $N_{p,i}$ value after reconstruction if $decision = true$; the current $N_{p,i}$ value, otherwise.

if $N > K$ **then**

Perform peer selection based on balance.

Let q be the selected peer.

end if

if $q \neq p$ **then**

Request q to perform reconstruction.

end if

if $decision = true \wedge q = p$ **then**

Perform reconstruction.

end if

end for

Figure 4.4: Algorithm Outline

the peers adjacent to the end peers of $L_{p,i}$.

3. Choose reconstruction, that is, moving to the other list at level $i+1$, if link delay estimated in Step 2 is less than the current link delay measured in Step 1.

Step 2 requires message traversals through $L_{p,i}$ in both directions. Thus, in this step, p can know $L_{p,i}$, and in turn, the current value of $N_{p,i}$ and its new value if reconstruction occurs. Let N be the new $N_{p,i}$ value if reconstruction is chosen; otherwise, let N be its current value.

Peer Selection Based on Balance

The topological balance is considered if $N > K$, where K is a design parameter of the algorithm. Note that large N implies some topological imbalance; thus, K can be considered a threshold specifying the permissible degree of imbalance. The idea here is to select one peer in $L_{p,i}$ and request the selected peer to move to the other list at level $i+1$.

1. Request that each peer in $L_{p,i}$ measures the current link delay at level $i+1$ and estimates its value when that peer moves to the other list. Each peer sends back to p the difference between these two values, that is, the estimated delay minus the current delay.

2. Select the peer that returned the lowest value.

Note that the selected peer can be p itself. If this is the case and the decision to move has already been made on the basis of proximity, that decision is simply cancelled. If the selected peer is not p, then p requests the selected peer to move to the different list at level $i+1$.

Reconstruction Operation

A peer p performs reconstruction by changing the list to which it belongs at level $i + 1$. This operation is invoked when p chooses it on the basis of proximity or is requested by another peer in $L_{p,i}$.

4.4 Required Messages

In this section, I investigate the number of required messages in the proposed skip graph for each algorithm. I assume that each peer executes topology reconstruction enough times, and all $N_{p,i}$ satisfy $N_{p,i} \leq K$ for any peer p and level i. The value K is a previously chosen constant and is sufficiently smaller than the number of peers. I denote the number of peers as n_0.

The number of messages required for each process depends on the height of the graph. Therefore, I first demonstrate that the height of the proposed skip graph is $O(\log n_0)$ and next demonstrate that the number of required messages for the algorithms is also $O(\log n_0)$.

[Lemma 1] At least one peer in a list joins a different list from the other peers at the upper levels.

[Proof] If the number of peers in a list of level i is greater than K, the lemma follows obviously because $N_{p,i} \leq K$.

Now, suppose that the number of peers in a list at level i is smaller than K, and all peers in this list belong to the same list at level $i + 1$ at a particular moment. In this case, when one of these peers starts the reconstruction operation, link delay is 0 in the other list at level $i + 1$, and $N_{p,i}$ must be K or less. Therefore, the peer initiating the reconstruction operation must change the list at level $i + 1$. If only

one peer belongs to the list at level i, the maximum level of the peer is i.

Hence, at least one peer in a list joins a different list from the other peers at the upper level.

[Theorem 1] The height of the proposed skip graph with n_0 peers is $O(\log n_0)$.

[Proof] For any peer p, I denote the number of peers in the list that p belongs to at level i as n_i.

(1) When $n_i > K+1$, n_{i+1} is maximised if from the end of the list at level i, K peers join one list at level $i+1$ and the next peer joins the other list and another K peers join the list and so on. In this case, $n_{i+1} = (\frac{K}{K+1}) * (n_i + 1)$, and $n_{i+1} \leq (\frac{K}{K+1}) * (n_i + 1)$ in any part of the graph. Therefore, $n_r \leq n_0 * (\frac{K}{K+1})^r + K$ at any level r.

The minimum r that satisfies $n_r \leq K+1$ is less than s that satisfies $n_0 * (\frac{K}{K+1})^s + K = K+1$, because $s = \log_{\frac{K+1}{K}} n_0$, $r \leq \log_{\frac{K+1}{K}} n_0$•

(2) When $n_i \leq K+1$, from Lemma 1, $n_{i+1} \leq n_i - 1$. Therefore, r' with $n_{i+r'} = 1$ satisfies $r' \leq K$

The maximum level of p is the sum of r and r' therefore, $r + r' \leq \log_{\frac{K+1}{K}} n_0 + K$• Because K is a constant, the maximum level of peer p is $O(\log n_0)$. Therefore, the height of the proposed skip graph is $O(\log n_0)$.

4.4.1 Searching Operation

To search for a key, a peer starts the searching operation from the maximum level. The initiating peer creates a search query consisting of the IP address and the target key and forwards the query to itself at the maximum level.

[Lemma 2] A search query is transferred at most $K-1$ times at the same level.

[Proof] Because $N_{p,i}$ satisfies $N_{p,i} \leq K$ for any p and i, a neighbour peer p' of p at level $i+1$ exists within $K+1$ hops from p in the list at level i. If p has to send a search query to p', query forwarding is executed at level $i+1$. Therefore, the sending of a search query is done fewer than $K+1$ times.

[Theorem 2] The searching operation requires query forwarding $O(\log n_0)$ times.

[Proof] From Theorem 1 and Lemma 2, the maximum number of times a search query is sent is calculated by $(\log_{\frac{K+1}{K}} n_0 + K - 1) * K = O(\log n_0)$.

4.4.2 Peer Joining

When a peer p joins the skip graph, it has one key.

1. Ask a peer that belongs to the skip graph to search for the key of p. Then, get the address of peer q that has the nearest key to p.

2. Send a join message to q. Set the links to join the list at level 0.

3. Choose the lists to join at levels above level 1. The lists are selected by topology reconstruction. Continue this operation until only peer p remains in the list.

[Theorem 3] The number of messages required to join the skip graph is $O(\log n_0)$.

[Proof] The joining operation contains the searching operation and list reconstruction. The searching operation requires $O(\log n_0)$ messages, as shown in Theorem 2. The number of messages required to reconstruct the list at level i is less than $K+1$. List reconstruction is executed from level 0 to the maximum level, and the maximum level is $O(\log n_0)$, as shown in Theorem 1. Therefore, the number of messages required to join the skip graph is $O(\log n_0)$.

4.4.3 Peer Removal

A peer is permitted to execute the removal operation at any time. This operation is executed as a normal removal operation or as part of topology reconstruction. When this operation is executed by peer p, a neighbouring peer initiates the reconstruction of the skip graph because $N_{p,i}$ may not satisfy $N_{p,i} \leq K$.

1. The departing peer sends a removal message to its right and left peers in the list. This message consists of the addresses of the peers so that they can set a new link.

2. The two peers that have received the removal message communicate with each other to construct a link. These peers send a completion message to the departing peer after setting the new link.

3. The departing peer cuts links at the level after receiving the completion message.

4. The departing peer executes this procedure from the maximum level to level 0.

[Theorem 4] The number of messages required for normal removal is $O(\log n_0)$.

[Proof] In the removal method at one level, the departing peer communicates only with its right and left neighbour peers in the list. Because the maximum level is $O(\log n_0)$, the number of messages required for the removal operation is also $O(\log n_0)$.

4.4.4 Topology Reconstruction

I define topology reconstruction as a set of reconstruction operations executed by one peer from level 0 to the maximum level.

[Theorem 5] When $K = 2$, the number of messages required for topology reconstruction is $O(\log n_0)$.

[Proof] When peer p starts the reconstruction operation, it sends messages to its right and left peers in the list at level i to check whether p has to migrate from a list during reconstruction. The message is sent to a peer that belongs to the same list as p at level i. When the peer belongs to a different list from p, the peer returns a receipt message to p. Therefore, the message is transported fewer than $K + 2$ times at each level. Because the number of required messages is also $O(K)$ in both decision-making procedures, the reconstruction operation at level i is completed in O(1). Since reconstruction operation is executed from level 0 to the maximum level, the number of messages required to finish the topology reconstruction is $O(\log n_0)$.

4.5 Application to Skip Graph Variants

The desirable properties of skip graphs have led to the development of interesting variants including skip B-trees [3], rainbow skip graphs [8], SkipNets [10], SkipIndex [29] and Znet [23]. In this section, I show how my approach can be incorporated into these skip graph variants. In particular, I consider skip B-trees and rainbow skip graphs.

4.5.1 Skip B-tree

By combining the advantages of skip graphs with B-trees, the skip B-tree provides efficient search operations. It has the following features:

- Each list is divided into sub-lists called blocks. The maximum number of peers in one block is $b = |\Sigma|$. Unlike the case in skip graphs, $|\Sigma|$ must be greater than two.
- Peers in the same block do not join the same list at upper levels. That is, for any two peers p and q in the same block at level i, $m(p)|i+1$ differs from $m(q)|i+1$, where $m|l$ denotes the lth element of a membership vector m.
- Each peer has two types of links: links to neighbour peers at each level and those to the neighbour blocks at each level.
- Each peer knows the maximum and minimum keys of the block.
- A search query is forwarded from block to block.

These features allow key searches with smaller hops than in a normal skip graph, because a query can bypass the peers in the same block. The second feature also allows the topology to remain balanced; thus, I focus on using my approach only to improve the proximity of neighbour peers.

Peer p initiates topology reconstruction at level i, and the level is increased one by one. The steps involved at each level are as follows:

1. Randomly select one symbol that is not used as the $i+1$th element of the membership vector of any peer in the block.

2. Send messages to the right and left blocks to search for peers with a membership vector containing the selected symbol as the first bit.

3. Check the link delays with the peers thus located. When the sum of these values is less than that of link delays with neighbours of peer p at level $i+1$, change the symbol of the $i+1$th element of the membership vector, and change the links to the located peers at level $i+1$.

4. When the reconnection is executed, perform the removal operation at all upper levels.

5. Choose the peers from the right and left blocks with minimal link delays.

6. Execute the same operations at the upper levels.

4.5.2 Rainbow Skip Graph

The rainbow skip graph is another variant of the skip graph. A notable feature is that peers have fewer links than in the skip graph. This results in shorter search path length.

The rainbow skip graph has the following features.

- All peers are divided into groups that are ordered entirely according to keys. The groups comprise a skip graph in which a group serves the same role as a single peer in a normal skip graph. The number of peers in a group depends on the height of this skip graph; it is greater than the height and less than twice the height. In a rainbow skip graph, $b = |\Sigma| = 2$.

- Each peer knows the maximum and minimum keys of the group.

- Each level is assigned to exactly one peer in a group, and that peer has links to the peers at that level in the left and right neighbour groups.

A problem with the rainbow skip graph is that searching requires more steps than in a normal skip graph, because peers must send a search query to another peer in the same group to change the level.

The rainbow skip graph has two lists: a vertical list in the group to connect between levels and a horizontal list to connect groups. Therefore, when applying my approach, peers first reconstruct the vertical list on the basis of proximity and then reconstruct the skip graph on the basis of balance and proximity. The vertical list consists all of the peers in the group. The peers in the vertical list are sorted in order of the levels to which they are assigned. If the number of peers in the group exceeds the maximum level of the group, some peers do not belong to the horizontal list.

When a peer p initiates topology reconstruction, it reconstructs the vertical list first, as follows

1. Choose one peer in the group that belongs to both vertical and horizontal lists.

2. Check the change in link delay of the vertical list that would occur if p and the chosen peer exchange positions in the vertical list.

3. If link delay decreases, p and the chosen peer exchange their positions by exchanging their links in the vertical and horizontal lists.

Second, p reconstructs the skip graph on the basis of balance and proximity. This part is the same as in the proposed construction of the skip graph in Section 4.3.2.

4.6 Simulations

In this section, I evaluate the proposed skip graph by comparing it with others. The simulation examines normal skip graph, proposed skip graph, skip B-tree, rainbow skip graph, advanced skip B-tree and rainbow skip graph that apply my proposed method. I call these graphs n-Graph, p-Graph, B-Tree, Rainbow, p-B-Tree and p-Rainbow, respectively. I set b, which indicates the number of lists one list at level i divided into that at level $i+1$, to 6 in B-Tree and p-B-Tree, and to 2 in the other graphs.

I use two measures to evaluate the efficiency of the graphs: search delay and search path length. Search delay is the sum of link delays to finish one searching operation, and the search path length is the translated times of the search query.

In the simulations, I use the transit-stub model as the physical network model [28]. I assume that there are 100 transit domains and that each transit domain has 100 stub domains. The delay time of the physical links is set to 10ms for the links between transit domains and 1ms for other links.

For a given set of parameter values, I perform 10 runs and average the results obtained. In each run, I construct a new physical network model and a new overlay topology.

4.6.1 Effect of K

Here, I evaluate how K, a design parameter of my proposed algorithm, affects the properties of the resulting skip graphs. This parameter is used to determine whether topology reconstruction occurs: if $N_{p,i} > K$ holds, peer p perceives a structural imbalance and decides to move to a different list. I measure search

delay and search path length for different values of K. I define search delay as the sum of link delay of all links that a query message traverses. I also define search path length as the number of peers through which a query message travels (including the destination peer). I set the number of peers n to 8000.

Figures 4.5 and 4.6 show search path length and search delay, respectively, versus K. As stated in the previous section, the parameter K serves as the threshold below which topological imbalance is permitted. Hence, as shown in Figure 4.5, as K increases, search path length also increases. The increase in search delay exhibited in Figure 4.6 can be accounted for by the increase in search path length.

Figure 4.7 shows the relationship between balance and the number of times reconstruction is executed. The average number of times of reconstructions per peer is depicted on the horizontal axis, and the total number of sub-lists $L_{p,i}$ such that $N_{p,i} = |L_{p,i}| > K$ for each peer p and level i is plotted on the vertical axis. This figure shows that if $K \geq 3$, only a few executions of reconstruction by a peer suffice to enforce a desired balance on the skip graph. On the other hand, when $K = 2$, it is much harder to satisfy the balance requirement, which may lead to numerous repeated executions of the reconstruction algorithm. Figure 4.6 shows that search delay is almost the same from $K = 2$ to 4. Hence, I set $K = 3$ in the following simulations.

4.6.2 Comparison of Search Delay and Search Path Length

In this simulation, I evaluate search delay and search path length. I let each peer search for each existing key. The results are averaged on all peer?key pairs. For p-Graph and p-Rainbow, parameter K, which is used for balance adjustment in these methods, is set to 3.

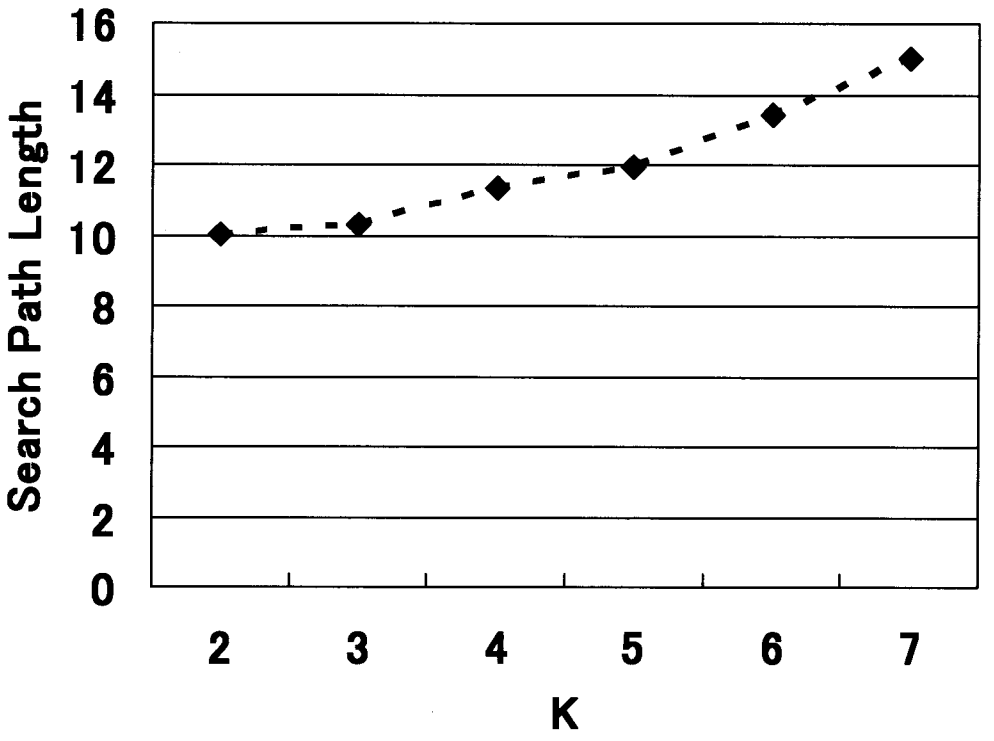

Figure 4.5: Search Path Length as a Function of K

The results shown in Figures 4.8 and 4.9 demonstrate that for normal skip graph, skip B-tree and rainbow skip graph, my proposed method reduces search delay by 20% to 30% while maintaining almost the same search path length. This shows that my proposed method can work well not only with the normal skip graph but also with its variants.

The results displayed in Figures 4.8 and 4.9 show that for normal skip graphs, skip B-trees, and rainbow skip graphs, the proposed method does not increase the number of hops, and it achieves a 20% to 30% reduction in search time.

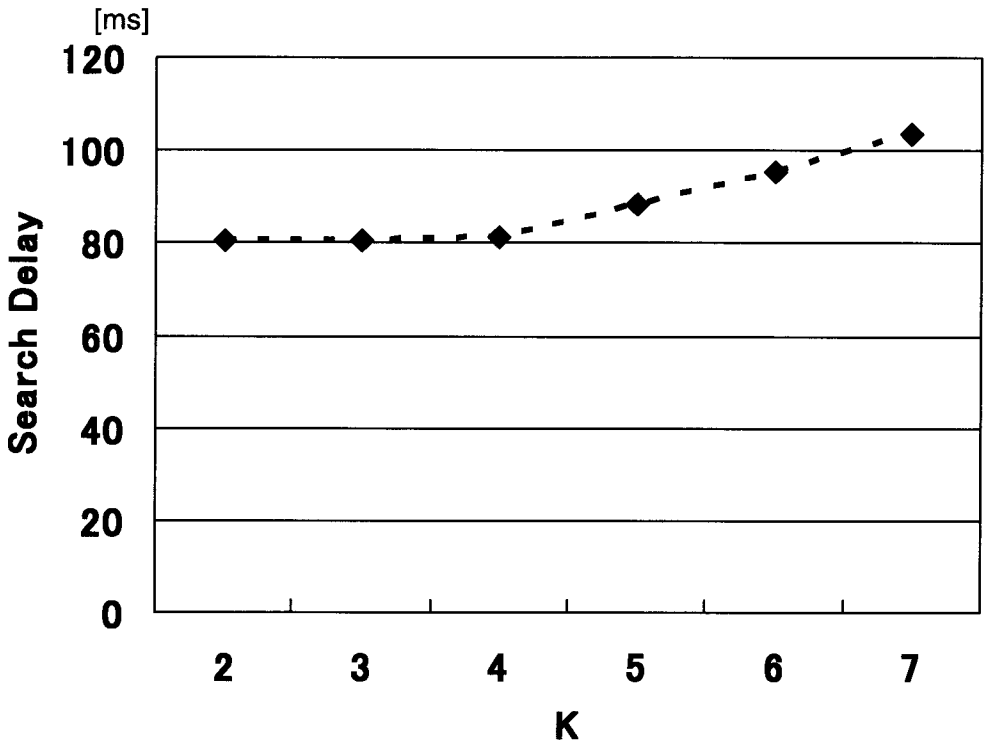

Figure 4.6: Search Delay as a Function of K

For rainbow skip graphs, the proposed method reduces search time more than those for the other graphs. This result stems from the group structure of a rainbow skip graph in which some peers constitute a group that behaves as if it were a single peer in a normal skip graph. In a group, a peer serves the role that a peer of a normal skip graph plays at a particular level. This reduces the degree, that is, the number of adjacent peers. On the other hand, this structure incurs additional communication overhead when a search descends to lower levels, in which case the search query needs to travel from one peer to another in the same group.

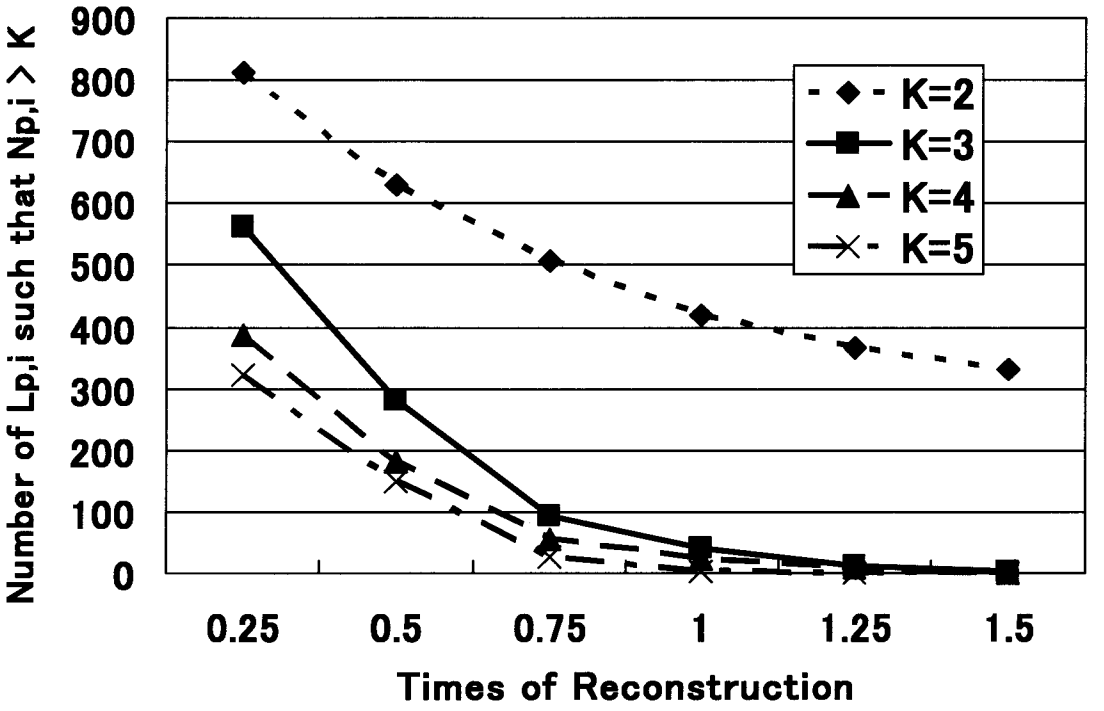

Figure 4.7: Relationship between Balance and Number of Times Reconstruction is Executed

Applying the proposed approach to a rainbow skip graph optimises both the links between peers in a group and those between groups. This results in a relatively high reduction in search time.

4.7 Discussion

Skip graph is a distributed network based on the skip list [18]. SkipNet [10] is another method for constructing an overlay network based on the skip list. More-

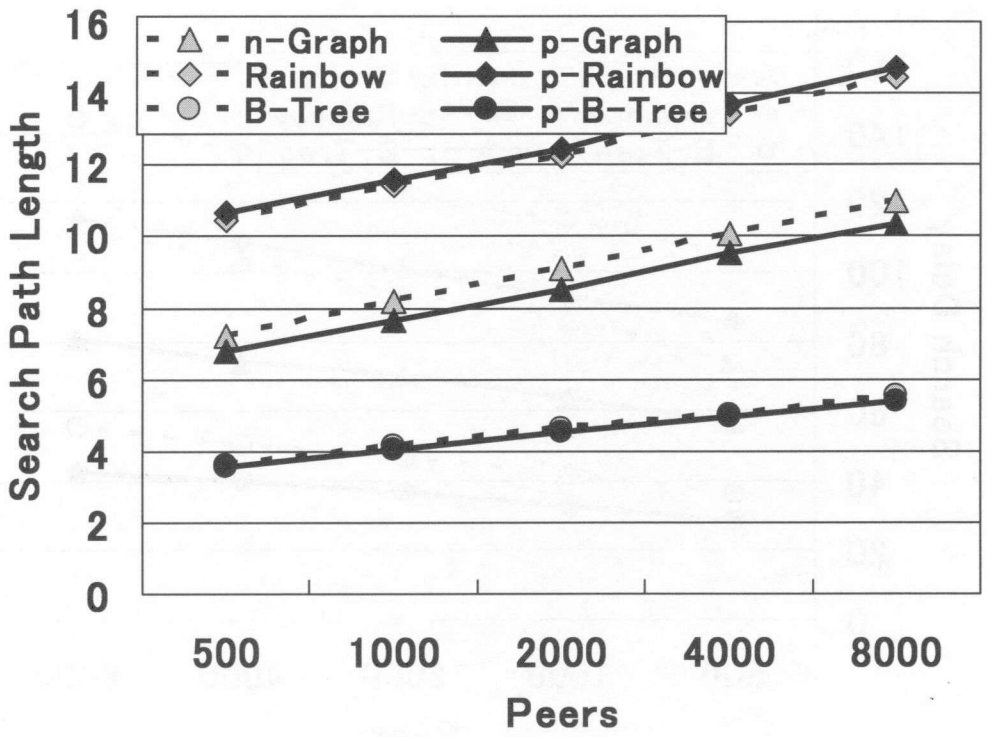

Figure 4.8: Comparison of Search Path Length

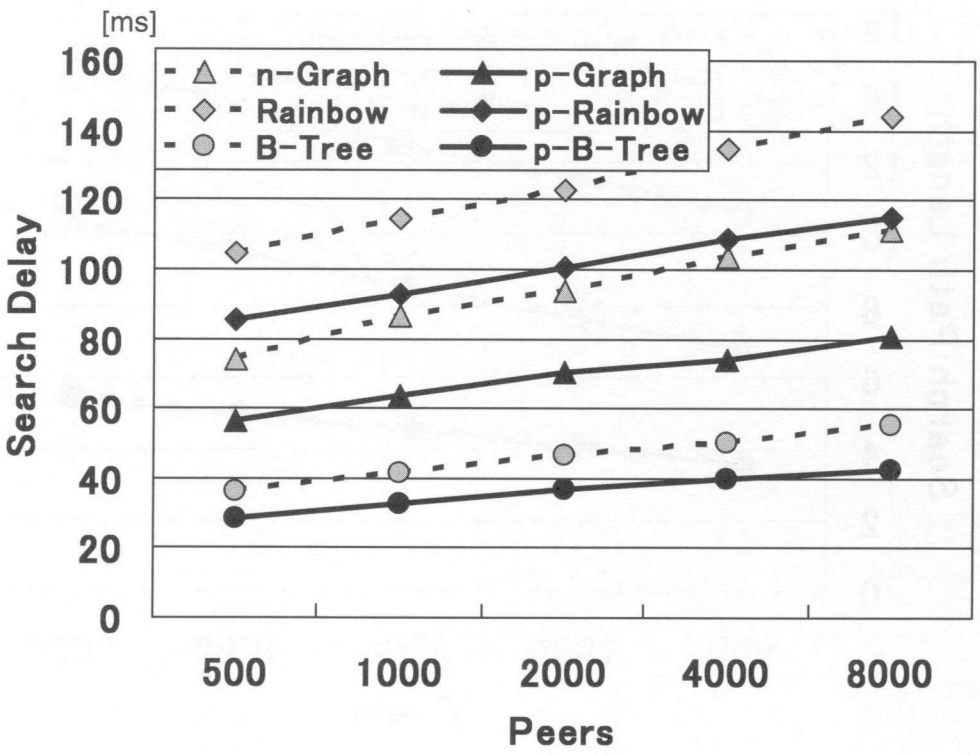

Figure 4.9: Comparison of Search Delay

over, variants of the skip graph exist in addition to skip B-tree and rainbow skip graph.

SkipNet consists of levels, similar to a skip graph. Each level consists of rings constructed by peers. In [10], two types of SkipNet are proposed: perfect SkipNet and probabilistic SkipNet. In a perfect SkipNet, all rings are perfectly balanced. However, the maintenance costs are very high. Therefore, the probabilistic SkipNet, in which each peer randomly chooses rings at each level, is more practical. I expect that my proposed approach can also reduce search delay when applied to the probabilistic SkipNet.

The strong rainbow skip graph proposed in [8] is an improvement in the rainbow skip graph. In this method, the graph topology is stringently controlled. This method is identical to my proposed method with $K = 1$, where K is the maximum value of Np, i. Because of the restricted graph topology, my proposed approach cannot reconstruct the horizontal lists. However, the proposed approach can reconstruct the vertical lists. Therefore, it can reduce search delay.

In a multi-key skip graph, one physical peer has one or more data items. Each physical peer creates some virtual peers. Each virtual peer has only one data item, and the virtual peers compose the skip graph. When a peer searches for a key, if it uses the search operation of the original skip graph, a physical peer may receive one search query several times. The multi-range forward method solves this problem [11]. The construction of a multi-key skip graph is the same as that of the original skip graph. Therefore, my proposed approach can reduce search delay in the multi-key skip graph as it does in the original skip graph.

4.8 Summary

In this chapter I proposed a method for constructing the topology of a skip graph. In this method, a peer checks for the possibility of reducing the search delay between its adjacent neighbours and that of improving the structural balance of the network topology. If the peer finds that possibility, it locally reshapes the topology by changing the list to which it belongs. Using the simulation results, I demonstrated that by considering both proximity (that is, link delays between neighbours) and the balance of the topology in network construction, search delay can be reduced substantially.

Chapter 5

Conclusion

5.1 Achievements

In this dissertation, I proposed approaches for reducing search delay in both unstructured and structured overlay networks.

The first achievement is to propose an algorithm for constructing unstructured overlays. My proposed method can construct overlay networks with low search delay, low path length, low clustering coefficient and high reachability. Therefore, in the overlay network constructed by my approach, peers can search for a target peer more rapidly and reliably than in those constructed by the existing methods.

The second achievement is to propose an algorithm for constructing skip graphs. I proposed an approach for constructing balanced and proximity-aware skip graphs. In the proposed algorithm, each peer chooses links so as to reduce search delay by measuring the balance of the graph and the link delay between peers. In the overlay network constructed by my proposed approach, peers can search for a key with a search delay that is about 30% lower than that in the original skip graph.

These achievements contribute significantly to the efficient use of P2P distributed systems because data search is one of the most important and frequently used functions in distributed network systems. My achievements contribute especially to systems with numerous peers. The number of peers that join a system typically increases as the performance of machines and networks improves. Therefore, the importance of my proposed approach will also increase.

5.2 Future Research

A possible research direction is to evaluate the effectiveness of the proposed approaches in dynamic environments. In the presence of high churn or fluctuation in communication time, each peer should execute the reconstruction operation frequently. It is worthwhile to investigate the relationship between network instability and the required frequency of reconstruction.

Bibliography

[1] P2P earthquake announcement. http://www11.plala.or.jp/taknet/p2pquake/

[2] J. Aspnes and G. Shah, Skip graphs. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, January 2003.

[3] I. Abraham, J. Aspnes, and J. Yuan, Skip B-trees. *Principles of Distributed Systems; 9th International Conference, OPODIS 2005; Pisa, Italy; December 2005; Revised Selected Papers*, Vol.3974, pages 366–380, December 2005.

[4] A.L. Barabási and R. Albert, Emergence of scaling in random networks. *Science*, Vol.286, No.5439, pages 509–512, October 1999.

[5] H. Cai and J. Wang, Exploiting geographical and temporal locality to boost search efficiency in peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems*, Vol.17, pages 1189–1203, October 2006.

[6] K.T. Chen and J.K. Lou, Toward an understanding of the processing delay of peer-to-peer relay nodes. In *Proc. International Conference on Dependable Systems and Networks*, pages 410–419, USA, June 2008.

[7] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, Vol.2009, pages 46–66, 2001.

[8] M.T. Goodrich, M.J. Nelson, and J.Z. Sun, The Rainbow Skip Graph: A fault-tolerant constant-degree distributed data structure. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA '06)*, pages 384–393, January 2006.

[9] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, The impact of DHT routing geometry on resilience and proximity. In *Proc. 2003 conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 381–194, August 2003.

[10] N.J.A. Harvey, J. Dunagan, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman, SkipNet: A scalable overlay network with practical locality properties. In *Proc. 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 113–126, March 2003.

[11] Y. Konishi, M. Yoshida, Y. Teranishi, K. Harumoto, and S. Shimojo, A proposal of a multi-key extension of skip graph. *IPSJ SIG Notes*, Vol.2007, No.58, pages 25–30, June 2007.

[12] J. Leitão, J. Pereira, and L. Rodrigues, Topology aware gossip overlays. *Technical Report 36, INESC-ID(2008)*, January 2008.

[13] J. Leitão, N.A. Carvalho, J. Pereira, R. Oliveira, and L. Rodrigues, On adding structure to unstructured overlay networks. *Handbook of Peer-to-Peer Networking*, Part.3, pages 327–365, February 2010.

[14] L.S. Liu and R. Zimmermann, Adaptive low-latency peer-to-peer streaming and its application. *Multimedia Systems*, Vol.11, No.6, pages 497–512, April 2006.

[15] Y. Liu, L. Xiao, X. Liu, L.M. Ni, and X. Zhang, Location awareness in unstructured peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems*, Vol.16, pages 163–174, February 2005.

[16] F. Makikawa, T. Matsuo, T. Tsuchiya, and T. Kikuno, Constructing overlay networks with low link costs and short paths. In *Proc. 6th IEEE International Symposium on Network Computing and Applications (IEEE NCA07)*, pages 299–304, July 2007.

[17] L. Massoulié, A.M. Kermarrec, and A.J. Ganesh, Network awareness and failure resilience in self-organising overlay networks In *Proc. 22nd IEEE Symposium on Reliable Distributed Systems (SRDS '03)*, pages 47–55, October 2003.

[18] W. Pugh, Skip lists: A probabilistic alternative to balanced trees. In *Communications of the ACM*, Vol.33, pages 668–676, June 1990.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, A scalable content-addressable network. In *Proc. 2001 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*, pages 161–172, August 2001.

[20] Y. Ren, C. Sha, W. Qian, A. Zhou, B. Ooi, and K. Tan, Explore the "small world phenomena" in pure P2P information sharing systems. In *Proc. the*

3rd International Symposium on Cluster Computing and the Grid, pages 232–239, May 2003.

[21] 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, pages 50–57, September 2002.

[22] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, November 2001.

[23] Y. Shu, B.C. Ooi, K.-L. Tan, and A. Zhou, Supporting multi-dimensional range queries in peer-to-peer systems. In *Proc. 5th IEEE International Conference on Peer-to-Peer Computing (P2P '05)*, pages 173-?180, September 2005.

[24] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, Chord: A scalable peer-To-peer lookup service for Internet applications. In *Proc. 2001 ACM SIGCOMM Conference*, pages 149–160, August 2001.

[25] C. Tang, R.N. Chang, and C. Ward, GoCast: Gossip-enhanced overlay multicast for fast and dependable group communication. In *Proc. International Conference on Dependable Systems and Networks*, pages 140–149, June 2005.

[26] D. Watts and S. Strogatz, Collective dynamics of small-world networks. In *Nature*, Vol.393, pages 440–442, April 1998.

[27] R.H. Wouhaybi and A.T. Campbell, Phenix: Supporting resilient low-diameter peer-to-peer topologies. In *Proc. IEEE Infocom '04*, Vol.1, pages 108–119, April 2004.

[28] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee, How to model an Internetwork. In *Proc. IEEE Infocom '96*, Vol.2, pages 594–602, March 1996.

[29] C. Zhang, A. Krishnamurthy, and R.Y. Wang, Skipindex: Towards a scalable peer-to-peer index service for high dimensional data. *Technical Report TR-703-04*, May 2004.

[30] X.Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, A construction of locality-aware overlay network: mOverlay and its performance. *IEEE Journal on Selected Areas in Communications*, Vol.22, pages 18–28, January 2004.

[31] B.Y. Zhao, L. Huang, S.C. Rhea, J. Stribling. A.D. Joseph, and J.D. Kubiatowicz, Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, Vol.22, pages 41–53, January 2003.